



*Εθνικό Μετσόβιο Πολυτεχνείο*

*Δ.Π.Μ.Σ. Συστήματα Αυτοματισμού*

*Κατεύθυνση Β:*

*Συστήματα Αυτομάτου Ελέγχου και Ρομποτικής*

**Μεταπτυχιακό Μάθημα:**

*Αισθητήρες*

*Ομάδα 3*

*Τρίτη και Τέταρτη Εργαστηριακή Άσκηση*

*Εισαγωγή στο Arduino*

**Μέλη Ομάδας – Α.Μ.:**

*Γεώργιος Κασσαβετάκης – 02121203*

*Γεώργιος Κρομμύδας – 02121208*

*Λάμπης Παπακώστας – 02121211*

ΑΘΗΝΑ

2023

## 1. Παράδειγμα - LED Blink

Σε αυτό το παράδειγμα μας δόθηκε ο κώδικας ***blink.ino***. Σκοπός του συγκεκριμένου προγράμματος είναι να αναβοσβήνει το LED λαμπάκι που υπάρχει ενσωματωμένο στην πλακέτα του **Arduino UNO REV3** και βρίσκεται στο **pin D13** της πλακέτας.



Σχήμα 1. Πλακέτα Arduino UNO REV3

### 1.1 Ερώτημα 1

Σε αυτό το ερώτημα έχουμε να τροποποιήσουμε τον δοθέντα κώδικα έτσι ώστε το **LED** να παραμένει αναμμένο για 1.5 sec και σβηστό για 0.5 sec. Αναλυτικά μπορούμε να δούμε τροποποιημένο τον κώδικα στο σχήμα 2.

```
26 // the setup function runs once when you press reset or power the board
27 void setup() {
28   // initialize digital pin LED_BUILTIN as an output.
29   pinMode(LED_BUILTIN, OUTPUT);
30 }
31
32 // the loop function runs over and over again forever
33 void loop() {
34
35   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
36   delay(1500); // wait for a second
37   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
38   delay(500); // wait for a second
39 }
```

Σχήμα 2. Κώδικας .ino πρώτου ερωτήματος

Με την έναρξη της εκτέλεσης της συνάρτησης `loop()` ξεκινάει η λειτουργία του προγράμματος. Η συνάρτηση `digitalWrite()` θα δώσει σήμα στο **LED** ώστε να ανάψει. Αντίστοιχα θα δοθεί από την ίδια συνάρτηση σήμα για να σβήσει το **LED**. Τροποποιούμε το όρισμα των συναρτήσεων `delay()`. Για να παραμείνει το **LED** αναμμένο για 1.5 sec, το όρισμα θα αλλάξει από 1000 msec σε 1500 msec. Αντίστοιχα για να παραμείνει σβηστό για 0.5 sec, θα τροποποιηθεί το όρισμα από 1000 msec σε 500 msec.

## 1.2 Ερώτημα 2

Σε επέκταση του προηγούμενου ερωτήματος, τροποποιήθηκε ο κώδικας έτσι ώστε να εκπέμπεται από το **LED** η λέξη **SOS** σύμφωνα με τα σήματα *Morse*. Αναλυτικά στον πίνακα 1 παρουσιάζονται τα γράμματα που χρειαζόμαστε.

S	...
O	---
Παύση μεταξύ των συμβόλων (. ή -)	.
Παύση μεταξύ των γραμμάτων	-

Πίνακας 1. Σήματα Morse Γραμμάτων

Καθώς το γράμμα **S** χρειάζεται τρεις τελείες για να υλοποιηθεί, τότε θα πρέπει να αναβοσβήσει τρεις φορές. Οπότε αυτό που θα κάνουμε είναι να φτιάξουμε ένα **for loop** στο οποίο θα αναβοσβήνουμε το λαμπάκι κάθε 0.5 sec. Με χρήση των συναρτήσεων `digitalWrite()` θα δίνουμε σήμα στο λαμπάκι ώστε να παίρνει τιμή 1 και 0. Ενδιάμεσα από αυτές θα χρησιμοποιηθούν και δύο συναρτήσεις `delay()` με όρισμα τα 500 msec. Έτσι θα σχηματιστούν οι τρεις τελείες και το γράμμα **S**. Με παρόμοιο τρόπο συντάσσεται και το block που αντιστοιχεί στο γράμμα **O** με τη μόνη διαφορά ότι αλλάζει το όρισμα της συνάρτησης `delay()` στα 1500 msec. Επιπλέον, ανάμεσα από τα σύμβολα έχουν τοποθετηθεί συναρτήσεις `delay()` στα 500 msec, που αντιστοιχεί στην παύση των συμβόλων. Τέλος, μετά την εμφάνιση κάποιου γράμματος έχουμε πάλι καθυστέρηση

1500 msec. Στο σχήμα 3 μπορούμε να δούμε και το block κώδικα που αντιστοιχεί σε αυτή τη διαδικασία.

```
26 // the setup function runs once when you press reset or power the board
27 void setup() {
28   // initialize digital pin LED_BUILTIN as an output.
29   Serial.begin(9600);
30   pinMode(LED_BUILTIN, OUTPUT);
31 }
32
33 // the loop function runs over and over again forever
34 void loop() {
35
36   for (int i = 0; i < 3; i++) { // Create S letter
37     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
38     delay(500); // wait for a second
39     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
40     delay(500); // wait for a second
41   }
42
43   delay(1500);
44
45   for (int i = 0; i < 3; i++) { // Create O letter
46     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
47     delay(1500); // wait for a second
48     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
49     delay(500); // wait for a second
50   }
51
52   delay(1500);
53
54   for (int i = 0; i < 3; i++) { // Create S letter
55     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
56     delay(500); // wait for a second
57     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
58     delay(500); // wait for a second
59   }
60
61   delay(1500);
62 }
63 }
```

Σχήμα 3. Κώδικας .ino δεύτερου ερωτήματος

## 2. Παράδειγμα – Έλεγχος LED μέσω σειριακής επικοινωνίας

Σε αυτό το παράδειγμα μας δίνεται ο κώδικας *PhysicalPixel.ino*. Η λειτουργία του συγκεκριμένου προγράμματος είναι να ανάβει το **LED** της πλακέτας μέσω μιας σειριακής εντολής χρησιμοποιώντας το *Serial Monitor* του *IDE*.

### 2.1 Ερώτημα 1

Σε αυτό το ερώτημα είχαμε να τροποποιήσουμε τα *if loops* έτσι ώστε όταν διαβάζεται το επιθυμητό γράμμα από το *Serial Monitor*, να δίνει σήμα στο **LED** για να ανάβει και να σβήνει. Ήδη, έχει υλοποιηθεί αυτό το κομμάτι με τη χρήση των χαρακτήρων *ASCII 'H'* και *'L'*. Εμείς επεκτείνουμε τη λογική συνθήκη του *if* ώστε να δέχεται και τους χαρακτήρες *'h'* και *'l'*. Έτσι, προσθέτοντας την λογική πράξη *or* που συμβολίζεται με *||*, θα διαβάζονται

σειριακά και οι δύο χαρακτήρες. Αναλυτικά μπορούμε να δούμε τη διαδικασία στο σχήμα 4.

```
24 const int ledPin = 13; // the pin that the LED is attached to
25 int incomingByte;      // a variable to read incoming serial data into
26
27 void setup() {
28   // initialize serial communication:
29   Serial.begin(9600);
30   // initialize the LED pin as an output:
31   pinMode(ledPin, OUTPUT);
32 }
33
34 void loop() {
35   // see if there's incoming serial data:
36   if (Serial.available() > 0) {
37     // read the oldest byte in the serial buffer:
38     incomingByte = Serial.read();
39     // if it's a capital H (ASCII 72), turn on the LED:
40     if (incomingByte == 'H' || incomingByte == 'h') {
41       digitalWrite(ledPin, HIGH);
42     }
43     // if it's an L (ASCII 76) turn off the LED:
44     if (incomingByte == 'L' || incomingByte == 'l') {
45       digitalWrite(ledPin, LOW);
46     }
47   }
48 }
```

Σχήμα 4. Κώδικας .ino πρώτου ερωτήματος

Έτσι, δίνοντας τον χαρακτήρα **ASCII** από το **Serial Monitor**, θα διαβαστεί από την εντολή `Serial.read()` και θα αποθηκευτεί η τιμή στη μεταβλητή `incomingByte`. Μέσα στο `if loop` θα χρησιμοποιηθεί η συνάρτηση `digitalWrite()` ώστε να δώσει το απαραίτητο σήμα στο **LED**.

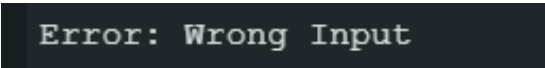
## 2.2 Ερώτημα 2

Σε αυτό το ερώτημα τροποποιήθηκε ο παραπάνω κώδικας έτσι ώστε κάθε φορά που διαβαστεί ένας χαρακτήρας **ASCII** από το **Serial Monitor** διαφορετικός από την επιθυμητή τετράδα ('H', 'L', 'h', 'l') να εμφανίζει ένα μήνυμα σφάλματος. Η διαδικασία αυτή φαίνεται αναλυτικά στο σχήμα 5.

```
34 void loop() {
35   // see if there's incoming serial data:
36   if (Serial.available() > 0) {
37     // read the oldest byte in the serial buffer:
38     incomingByte = Serial.read();
39     // if it's a capital H (ASCII 72), turn on the LED:
40     if (incomingByte == 'H' || incomingByte == 'h') {
41       digitalWrite(ledPin, HIGH);
42     }
43     // if it's an L (ASCII 76) turn off the LED:
44     else if (incomingByte == 'L' || incomingByte == 'l') {
45       digitalWrite(ledPin, LOW);
46     }
47     else {
48       Serial.println("Error: Wrong Input");
49     }
50   }
51 }
```

Σχήμα 5. Κώδικας .ino δεύτερου ερωτήματος

Τροποποιούμε το **if loop** έτσι ώστε να δημιουργηθεί μία δομή με **if** και **else if** και να κρατήσουμε την προηγούμενη δομή και κάθε φορά που διαβάζεται κάποιος άλλος χαρακτήρας, τότε θα εμφανίζει το μήνυμα που φαίνεται στο σχήμα 6. Για παράδειγμα, δόθηκε ο χαρακτήρας 'α' και παράχθηκε το **string** που σηματοδοτεί το σφάλμα.

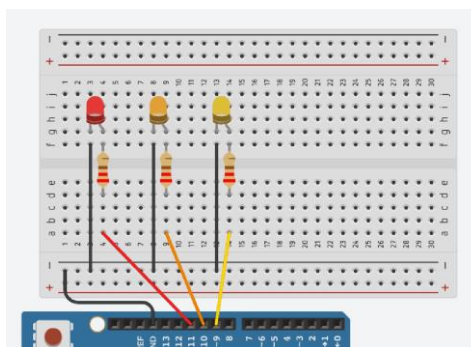


Error: Wrong Input

Σχήμα 6. Παράδειγμα εσφαλμένης εισόδου από το serial monitor

### 3. Παράδειγμα – Προσομοίωση φλόγας

Σε αυτό το παράδειγμα μας δόθηκε ο κώδικας **3\_LED\_Fire.ino**. Το συγκεκριμένο πρόγραμμα προσομοιώνει το φως που παράγει μια φωτιά. Αναλυτικά στο σχήμα 7 φαίνεται ο συνδεσμολογία του κυκλώματος. Παρατηρούμε ότι έχουν προστεθεί τρεις αντιστάσεις των 220 Ω. Οι αντιστάσεις αυτές χρειάζονται ώστε να υποβαθμιστεί η πτώση τάσης στο **LED**, έτσι ώστε να μην υπάρχει κίνδυνος να καεί. Ένα **LED** έχει τάση λειτουργίας από τα 1.6 V έως τα 3V. Τα λαμπάκια δέχονται τυχαίες τιμές κάθε φορά με τη χρήση της αναλογικής εισόδου από τη συνάρτηση *analogWrite()*.



Σχήμα 7. Συνδεσμολογία κυκλώματος προσομοίωσης φλόγας

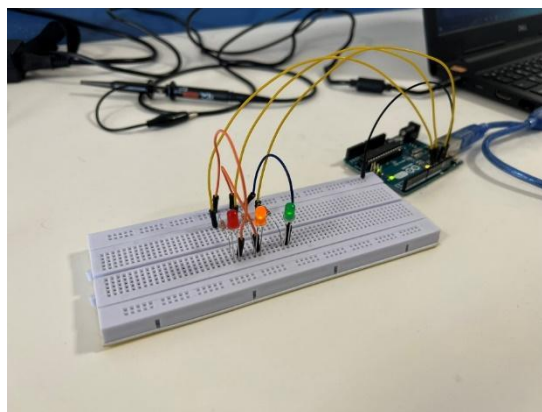
#### 3.1 Ερώτημα 1

Σε αυτό το ερώτημα είχαμε να τροποποιήσουμε τον κώδικα (βελτιστοποίηση) ώστε να μην αναγράφονται 3 φορές οι συναρτήσεις *pinMode()* και *analogWrite()*.

Δημιουργούμε δύο **for loops** στα οποία αναθέτουμε αντίστοιχα τα τιμές από το 9 έως το 11 που θα είναι οι εξοδοί του Arduino προς τα λαμπάκια. Είναι συνδεδεμένα σε PWM εξόδους για να μπορέσουν να πάρουν ενδιάμεσες τιμές και να παράγει η πλακέτα «αναλογικές τιμές». Αναλυτικά μπορούμε να δούμε τον κώδικα στο σχήμα 8. Επίσης, στο σχήμα 9 βλέπουμε και την πραγματική συνδεσμολογία του παραδείγματος.

```
1 int ledPin1 = 9; //Pin 1ου LED
2 int ledPin2 = 10; //Pin 2ου LED
3 int ledPin3 = 11; //Pin 3ου LED
4
5 void setup(){
6
7   Serial.begin(9600);
8   for(int mode = 9; mode <= 11; mode++){
9     pinMode(mode, OUTPUT);
10  }
11  // pinMode(ledPin1, OUTPUT); //Ορισμός ως Pin εξόδου
12  // pinMode(ledPin2, OUTPUT); //Ορισμός ως Pin εξόδου
13  // pinMode(ledPin3, OUTPUT); //Ορισμός ως Pin εξόδου
14  randomSeed(analogRead(A0)); //Λήψη τυχαιών αριθμών βάσει της εισόδου A0
15
16 }
17
18 void loop(){
19
20   for(int ledPin = 9; ledPin <= 11; ledPin++){
21     analogWrite(ledPin, random(0, 130) + 125);
22   }
23
24   // analogWrite(ledPin1, random(0, 130)); //Τυχαίο duty cycle μεταξύ των τιμών 125 & 255
25   // analogWrite(ledPin2, random(0, 130));
26   // analogWrite(ledPin3, random(0, 130));
27   delay(random(0, 120)); //Τυχαία καθυστέρηση μεταξύ 0 & 120 ms
28
29 }
```

Σχήμα 8. Κώδικας .ino πρώτου ερωτήματος

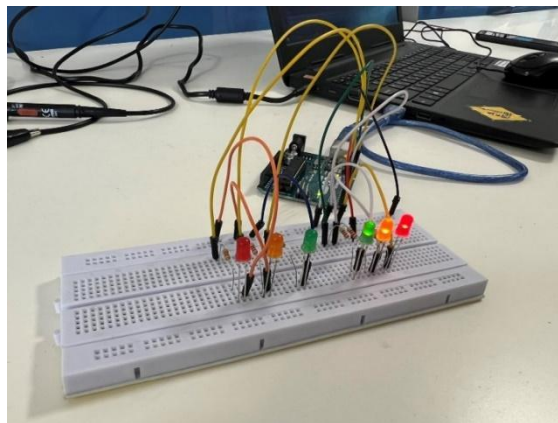


Σχήμα 9. Συνδεσμολογία κυκλώματος πρώτου ερωτήματος

## 3.2 Ερώτημα 2

Σε αυτό το ερώτημα ζητείται να επεκταθεί ο κώδικας για να διαχειρίζεται 6 **LEDs**. Αρχικά, τροποποιούμε τη συνδεσμολογία του κυκλώματος, προσθέτοντας 6 **LED** στο

**breadboard** μαζί με 6 αντιστάσεις των 220 Ω. Τα **LED** είναι όλα συνδεδεμένα σε **PWM(~)** εξόδους, έτσι ώστε να μπορούν να πάρουν και ενδιάμεσες τάσεις. Αναλυτικότερα φαίνεται το κύκλωμα στο σχήμα 10.



Σχήμα 10. Συνδεσμολογία κυκλώματος δεύτερου ερωτήματος

Το επόμενο βήμα είναι να τροποποιήσουμε τον κώδικα του προγράμματος, έτσι ώστε να λειτουργούν και τα 6 **LED**. Αρχικά ορίζουμε τα **pins** ως **int** τιμές {9,10,11,3,5,6} έτσι ώστε να χρησιμοποιήσουμε αυτές τις εξόδους του Arduino. Εν συνεχεία, δημιουργούμε μία στατική δομή δεδομένων (πίνακα), στον οποίο θα αποθηκεύσουμε τις μεταβλητές που περιέχουν την πληροφορία των εξόδων. Το επόμενο βήμα είναι να τροποποιήσουμε τα **for loops** έτσι ώστε να ορίζουμε τα **pinMode()** και τα **analogWrite()** για τα παραπάνω **pins**. Μέσα στις συνθήκες του **for loop** ορίζουμε έναν **index i** ο οποίος θα τρέξει όλον τον πίνακα. Έτσι σε κάθε επανάληψη θα αναθέτουμε σε κάθε **pin** την ιδιότητα της εξόδου. Το επόμενο βήμα είναι να του αναθέτουμε τυχαία μια τιμή ώστε να ανάβει το κάθε λαμπάκι με κάποια τάση. Έτσι, αντίστοιχα ορίζουμε και ένα **for loop** όπου θα δίνουμε σε κάθε έξοδο μια τυχαία τιμή από το σύνολο τιμών  $random(0,120) + 75$ . Το συγκεκριμένο διάστημα ορίστηκε, καθώς παρατηρήθηκε πως και τα 6 **LED** αναβοσβήνουν γρήγορα παίρνοντας διαφορετικές τάσεις. Αναλυτικότερα μπορούμε να δούμε τον κώδικα στο σχήμα 11.

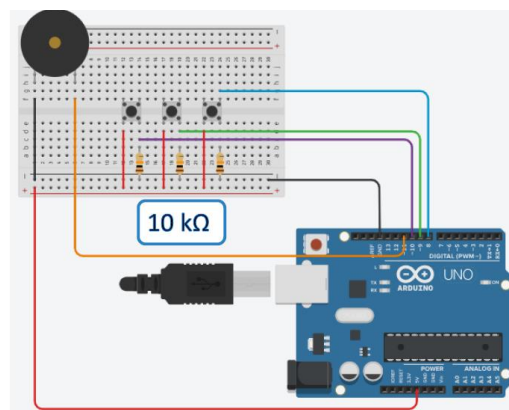


```
1 int ledPin1 = 9; //Pin 1ου LED
2 int ledPin2 = 10; //Pin 2ου LED
3 int ledPin3 = 11; //Pin 3ου LED
4 int ledPin4 = 3; //Pin 4ου LED
5 int ledPin5 = 5; //Pin 5ου LED
6 int ledPin6 = 6; //Pin 6ου LED
7
8 const int pinArray[6] = { ledPin1, ledPin2, ledPin3, ledPin4, ledPin5, ledPin6 };
9 void setup() {
10
11     Serial.begin(9600);
12     for (int i = 0; i < 6; i++) {
13         pinMode(pinArray[i], OUTPUT);
14     }
15     // pinMode(ledPin1, OUTPUT); //Ορισμός ως Pin εξόδου
16     // pinMode(ledPin2, OUTPUT); //Ορισμός ως Pin εξόδου
17     // pinMode(ledPin3, OUTPUT); //Ορισμός ως Pin εξόδου
18     randomSeed(analogRead(A0)); //Λήψη τυχαίων αριθμών βάσει της εισόδου A0
19 }
20
21 void loop() {
22
23     for (int i = 0; i < 6; i++) {
24         analogWrite(pinArray[i], random(0, 120) + 75);
25     }
26
27     // analogWrite(ledPin1, random(0, 130)); //Τυχαίο duty cycle μεταξύ των τιμών 125 & 255
28     // analogWrite(ledPin2, random(0, 130));
29     // analogWrite(ledPin3, random(0, 130));
30     delay(random(0, 120)); //Τυχαία καθυστέρηση μεταξύ 0 & 120 ms
31 }
```

Σχήμα 11. Κώδικας .ino δεύτερου ερωτήματος

## 5. Παράδειγμα – Piano

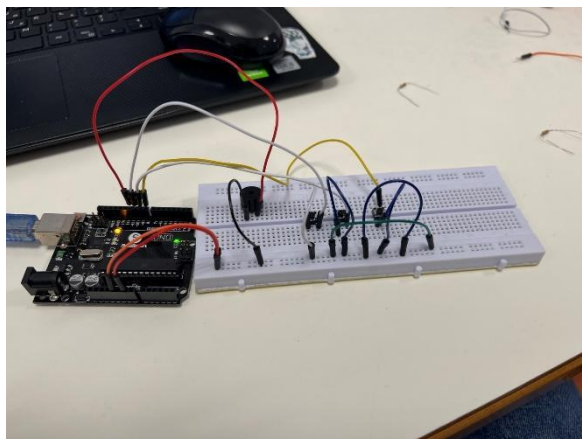
Σε αυτό το παράδειγμα μας δόθηκε ο κώδικας **Piano.ino**. Ο κώδικας αυτός αναπαράγει τρεις νότες με το πάτημα τριών κουμπιών. Οι νότες αυτές έχουν διαφορετικές συχνότητες. Κατά το πάτημα των κουμπιών αναπαράγεται ήχος μέσω ενός **buzzer**. Η συνδεσμολογία αυτού του κυκλώματος είναι η εξής:



Σχήμα 12. Κύκλωμα διάταξης πιάνου

### 5.1 Ερώτημα 1

Καθώς έχουν αφαιρεθεί οι αντιστάσεις από τα κουμπιά, θα προκύψει το νέο κύκλωμα που φαίνεται στο σχήμα 13. Παρατηρούμε πως έχει προκύψει κάποιο πρόβλημα.



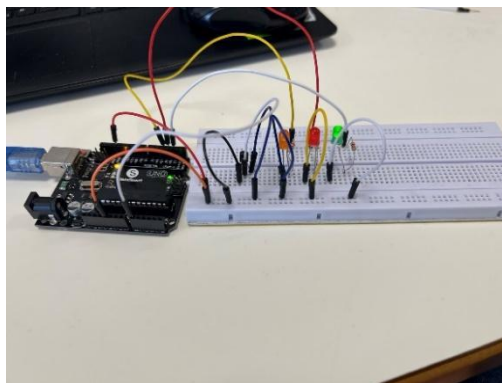
Σχήμα 13. Κύκλωμα διάταξης πιάνου χωρίς αντιστάσεις.

## 5.2 Ερώτημα 2

Κατά την αφαίρεση των αντιστάσεων προκύπτει θόρυβος στο **buzzer**, καθώς με το πάτημα κάποιου κουμπιού, εφαρμόζεται η αντίστοιχη συχνότητα. Αυτό οφείλεται στην πτώση τάσης στα κουμπιά της συνδεσμολογίας. Καθώς οι αντιστάσεις χρησιμοποιούνται για τη μείωση των ρευμάτων και του θορύβου στα **buttons**.

## 6. Παράδειγμα – Αισθητήρας Θερμοκρασίας TMP36

Σε αυτό το παράδειγμα μας δόθηκε ο κώδικας **TMP36.ino**. Χρησιμοποιείται ένας αισθητήρας θερμοκρασίας ο οποίος ανιχνεύει τη θερμοκρασία του χώρου. Το πρόγραμμα διαβάζει τα σήματα θερμοκρασίας του αισθητήρα και ανάβει το αντίστοιχο λαμπάκι το οποίο μας δείχνει το επίπεδο της θερμοκρασίας. Η συνδεσμολογία του κυκλώματος είναι η εξής:



Σχήμα 14. Κύκλωμα συνδεσμολογίας αισθητήρα θερμοκρασίας.

## 6.1 Ερώτημα 1

Σε αυτό το ερώτημα είχαμε να τροποποιήσουμε τον κώδικα έτσι ώστε κάθε φορά που διαβάζεται μια θερμοκρασία, να ανάβει ένα **LED**. Οι προδιαγραφές αυτές φαίνονται στον παρακάτω πίνακα.

Θερμοκρασία	LEDs
$\theta \leq 25$	Πράσινο
$25 < \theta \leq 35$	Πράσινο & Πορτοκαλί
$\theta > 35$	Πράσινο & Πορτοκαλί & Κόκκινο

Πίνακας 2. Αντιστοίχιση θερμοκρασιών και LED

Η τροποποίηση που θα κάνουμε στον κώδικα φαίνεται στο παρακάτω σχήμα.

```
void loop()
{
  int tempValue = analogRead(tempPin);
  float temp = tempValue * 5000.0 / 1024.0;
  float temp_C = (temp - 500) / 10;
  Serial.println(tempValue);
  Serial.print("Temp = ");
  Serial.print(temp_C);
  Serial.println("C");
  if (temp_C <= 25) {
    digitalWrite(green_pin, HIGH);
    digitalWrite(orange_pin, LOW);
    digitalWrite(red_pin, LOW);
  }
  else if (temp_C > 25 && temp_C <= 35) {
    digitalWrite(green_pin, HIGH);
    digitalWrite(orange_pin, HIGH);
    digitalWrite(red_pin, LOW);
  }
  else {
    digitalWrite(green_pin, HIGH);
    digitalWrite(orange_pin, HIGH);
    digitalWrite(red_pin, HIGH);
  }
  delay(1000);
}
```

Σχήμα 15. Κώδικας πρώτου ερωτήματος.

Κάθε φορά που διαβάζεται μια τιμή τάσης από το **tempPin** μέσω της εντολής **AnalogRead()**, αυτή, τότε υπολογίζεται η τάση εξόδου σύμφωνα με τον τύπο της δεύτερης εντολής του σχήματος. Εν συνεχεία υπολογίζεται η θερμοκρασία, σύμφωνα με την τάση που διάβασε ο μικροελεγκτής από τον αισθητήρα. Τέλος, κάθε θερμοκρασία που έχει υπολογιστεί, ελέγχεται σε ένα if loop. Εάν είναι μικρότερη ή ίση με τους 25°C, τότε θα πρέπει να ανάψει μόνο το πράσινο **LED**. Οπότε εισάγεται στον πρώτο βρόχο το πρόγραμμα και μέσω της εντολής **digitalWrite()** θα ανάψει μόνο το πρώτο **LED**. Στην περίπτωση που έχει διαβάσει μια θερμοκρασία μεταξύ των τιμών 25°C και 35°C, τότε θα εκτελεστεί το δεύτερο **loop**. Τότε θα ανάψει το πράσινο και πορτοκαλί **LED** μέσω των εντολών **digitalWrite()**. Σε περίπτωση που διαβαστεί θερμοκρασία μεγαλύτερη των

35°C, τότε αυτό που συμβαίνει είναι το πρόγραμμα να εισέρχεται στον τρίτο βρόχο και ανάβει και τα τρία **LED**, μέσω της εντολής *digitalWrite()*.

## 6.2 Ερώτημα 2

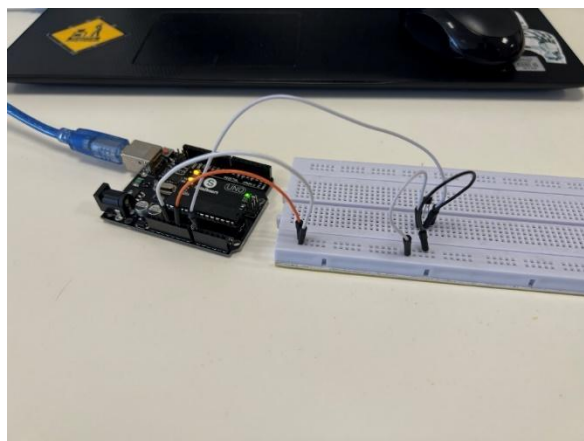
Στην περίπτωση που έχουμε τάση τροφοδοσίας  $V_{cc} = 3.3\text{ V}$  και η ανάλυση του **ADC** του **Arduino** είναι 12 bits, τότε θα έχουμε δείγματα  $2^{12} = 4096\text{ samples}$ . Έτσι, η εντολή που υπολογίζει την τάση εξόδου του αισθητήρα θα τροποποιηθεί ως εξής:

```
float temp = tempValue * 3300.0 / 4096.0;
```

Σχήμα 16. Τύπος υπολογισμού τάσης εξόδου.

## 7. Παράδειγμα – Hall SS49

Σε αυτό το παράδειγμα μας δίνεται ο κώδικας **7\_Hall\_SS49.ino**. Το πρόγραμμα αυτό είναι υπεύθυνο να δέχεται μετρήσεις από τον αισθητήρα Hall SS49, ώστε να ανιχνεύεται το μαγνητικό πεδίο ενός μαγνήτη. Η συνδεσμολογία του κυκλώματος φαίνεται στο σχήμα 17.



Σχήμα17. Συνδεσμολογία κυκλώματος αισθητήρα Hall SS49.

### 7.1 Ερώτημα 1

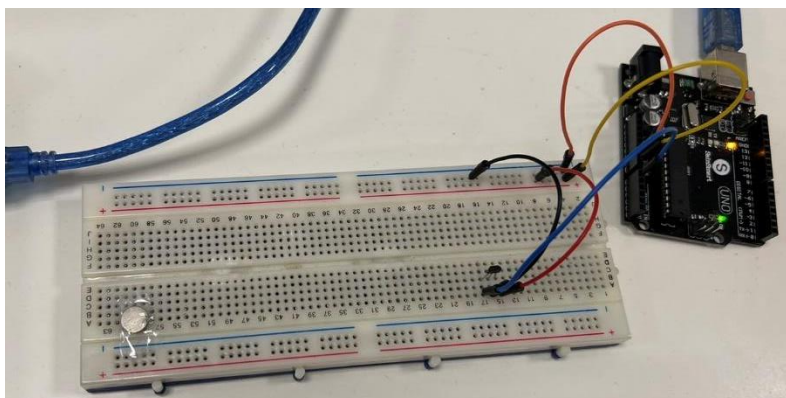
Σε αυτό το ερώτημα είχαμε να τροποποιήσουμε τον κώδικα που μας δόθηκε έτσι ώστε στην απουσία του μαγνήτη το πρόγραμμα να διαβάζει πεδίο 0 mT. Αυτό για να γίνει θα πρέπει να προσθέσουμε ένα *offset* στην τάση εισόδου που διαβάζει το Arduino.

Συγκεκριμένα, γνωρίζουμε πως το  $offset = 250mV$ , από την καμπύλη του datasheet του αισθητήρα Hall SS49. Όταν ο αισθητήρας παράξει αυτή την τάση, τότε γνωρίζουμε πως το μαγνητικό πεδίο είναι μηδενικός. Οπότε με την αρχικοποίησή του, που φαίνεται και στο σχήμα 18, μπορούμε να την εισάγουμε στην εντολή `map()` στο πρώτο πεδίο, έτσι ώστε με την μη ανίχνευση μαγνήτη να εμφανίζεται στο Serial Monitor η τιμή  $0mT$ . Πράγματι, στο σχήμα 18 παρατηρούμε πως με την απουσία μαγνήτη δεν ανιχνεύει μαγνητικό πεδίο. Ενώ με την παρουσία ανιχνεύει.

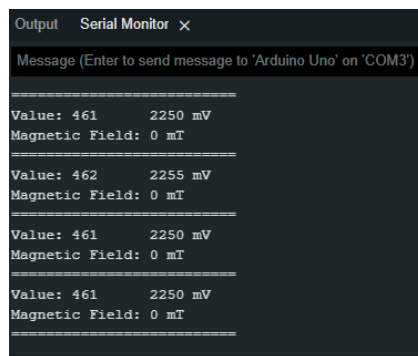
```
1  int pin = A0;           //Pin εισόδου αισθητήρα Hall
2  int vcc = 5000;         //Τάση τροφοδοσίας αισθητήρα Hall σε mV
3
4  int value;              //Μεταβλητή τιμής στο εύρος 0-1023
5  int voltage;            //Μεταβλητή τάσης στο εύρος 0-Vcc
6  int field;              //Μεταβλητή τιμής πεδίου σε mT
7  int offset = 250;       //Μεταβλητή τάσης offset mV
8  //int field_offset = -17; //Μεταβλητή τιμής πεδίου offset σε mT
9
10 void setup() {
11   Serial.begin(9600); //Έναρξη σειριακής επικοινωνίας
12 }
13
14 void loop() {
15   value = analogRead(pin); //Ανάγνωση τιμής του αισθητήρα Hall
16   voltage = value * (vcc / 1024.0); //Μετατροπή σε τιμή τάσης (mV)
17   field = map(voltage+offset, 1000, 4000, -100, 100); //Υπολογισμός μαγνητικού πεδίου (mT)
18
19   //Εμφάνιση αποτελεσμάτων
20   Serial.print("Value: ");
21   Serial.print(value);
22   Serial.print("\t");
23   Serial.print(voltage);
24   Serial.println(" mV");
25   Serial.print("Magnetic Field: ");
26   Serial.print(field);
27   Serial.println(" mT");
28   Serial.println("-----");
29   delay(1000); //Λήψη νέας τιμής κάθε 1sec
30 }
```

Σχήμα 18. Κώδικας πρώτου ερωτήματος.

Στο σχήμα 19 παρατηρούμε την συνδεσμολογία του κυκλώματος με την απουσία του μαγνήτη από τον αισθητήρα. Τα αποτελέσματα παρουσιάζονται και στο σχήμα 20. Όπου πράγματι με την απουσία του ο αισθητήρας διαβάζει  $0mT$ .

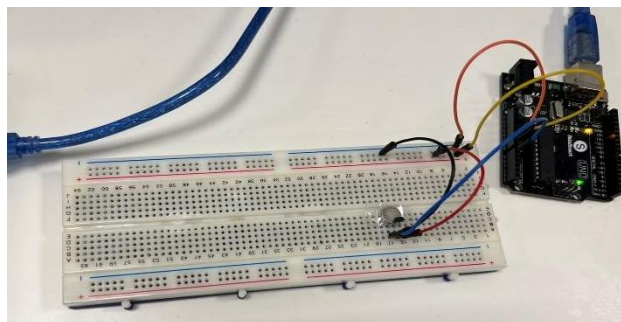


Σχήμα 19. Ανίχνευση πεδίου με απουσία μαγνήτη.

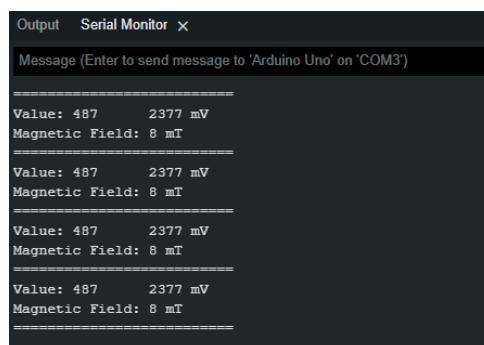


Σχήμα 20. Αποτέλεσμα αισθητήρα Hall με απουσία μαγνήτη.

Στο σχήμα 21 βλέπουμε την συνδεσμολογία του κυκλώματος με την παρουσία του μαγνήτη δίπλα στον αισθητήρα Hall. Στο σχήμα 22 παρατηρούμε το αποτέλεσμα της μέτρησης, όπου ανιχνεύεται μαγνητικό πεδίο με τιμή 8 mT.



Σχήμα 21. Ανίχνευση μαγνητικού πεδίου με παρουσία μαγνήτη.

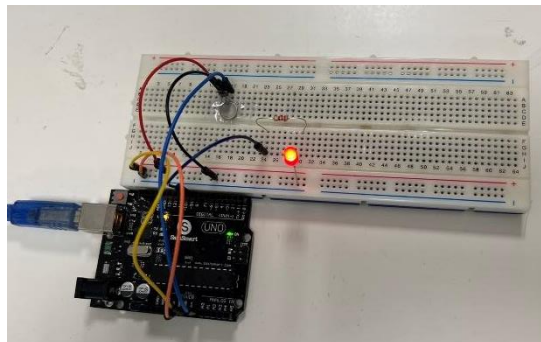


Σχήμα 22. Αποτέλεσμα αισθητήρα με παρουσία μαγνήτη.

## 7.2 Ερώτημα 2

Σε αυτό το ερώτημα τροποποιήθηκε και η συνδεσμολογία του κυκλώματος, μαζί με τον κώδικα. Προστέθηκε ένα *LED* και μία αντίσταση των  $R = 2.2\text{ k}\Omega$ . Το νέο κύκλωμα διακρίνεται και στο σχήμα 23. Με την χρήση του *LED* θα ανιχνεύουμε εάν υπάρχει μαγνητικό πεδίο. Η αντίσταση χρησιμοποιείται ως **pullup** αντίσταση για να υπάρχει

πτώση τάσης στο *LED* για να μην καταστραφεί. Επίσης, στο σχήμα βλέπουμε πως το λαμπάκι ανάβει όταν βρίσκεται κοντά ο μαγνήτης.



Σχήμα 23. Τροποποιημένη συνδεσμολογία κυκλώματος αισθητήρα Hall.

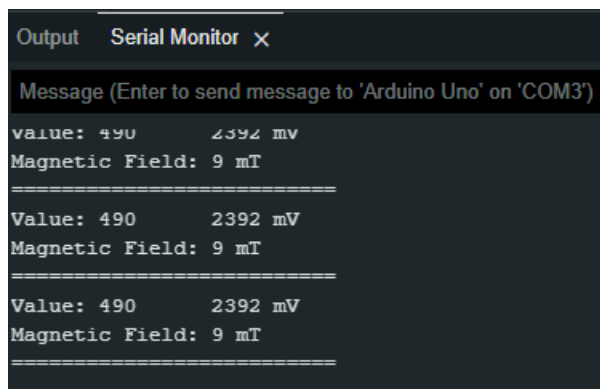
Για την επίτευξη της μέτρησης, τροποποιήθηκε ο κώδικας, έτσι ώστε κατά την απουσία του μαγνήτη να μην ανάβει το **LED**. Κατά την παρουσία του μαγνήτη θα ανάβει, με την προϋπόθεση πως το μαγνητικό πεδίο έχει κατά απόλυτη τιμή μεγαλύτερη του  $5\text{ mT}$ . Αρχικά, ορίστηκε το **pin** εισόδου από το 8 του **Arduino** που είναι και **PWM** είσοδος. Στην συνάρτηση `setup()` γίνεται η αρχικοποίησή του. Στη συνέχεια, στην συνάρτηση `loop()` δημιουργήθηκε ένα `if loop` το οποίο θα διαβάζει την τιμή του μαγνητικού πεδίου. Στην περίπτωση που η απόλυτη τιμή της μεταβλητής **field** (μαγνητικό πεδίο) έχει ξεπεράσει την τιμή των  $5\text{ mT}$ , τότε με την χρήση της συνάρτησης `digitalWrite()` θα δίνεται η τιμή **HIGH** στο **LED**, με αποτέλεσμα να ανάβει. Σε διαφορετική περίπτωση, πάλι με την χρήση της ίδιας συνάρτησης θα αποτυπώνεται η τιμή **LOW** στο **LED** στην περίπτωση που διαβαστεί μαγνητικό πεδίο στο διάστημα ( $-5\text{ mT}, 5\text{ mT}$ ).

```
1 //Μεταβλητή τάσης του κυκλώματος
2 int voltage;
3 //Μεταβλητή τιμής πεδίου σε mT
4 int field;
5 //Μεταβλητή τάσης offset mV
6 int offset = 250;
7 //Μεταβλητή τιμής πεδίου offset σε mT
8 int field_offset = -17;
9 int Led = 8;
10
11 void setup() {
12   Serial.begin(9600); //Έναρξη σειριακής επικοινωνίας
13   pinMode(Led, OUTPUT);
14 }
15
16 void loop() {
17   value = analogRead(pin); //Ανάγνωση τιμής του αισθητήρα Hall
18   voltage = value * (vcc / 1024.0); //Μετατροπή σε τιμή τάσης (mV)
19   field = map(voltage+offset, 1000, 4000, -100, 100); //Υπολογισμός μαγνητικού πεδίου (mT)
20
21   if(abs(field)>5){
22     digitalWrite(Led, HIGH);
23   }else{
24     digitalWrite(Led, LOW);
25   }
26
27   //Εμφάνιση αποτελεσμάτων
28   Serial.print("Value: ");
29   Serial.print(value);
30   Serial.print("\n");
31   Serial.print(voltage);
32   Serial.println(" mV");
33   Serial.print("Magnetic Field: ");
34   Serial.print(field);
35   Serial.println(" mT");
36   Serial.println("=====");
37   delay(1000); //Λήψη νέας τιμής κάθε 1sec
38 }
```

Σχήμα 24. Κώδικας δεύτερου ερωτήματος.

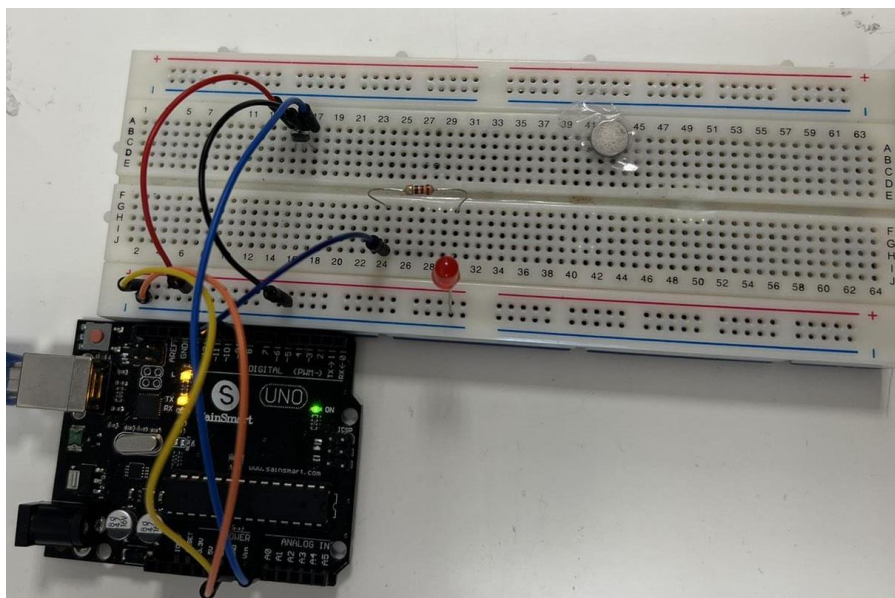


Παρακάτω μπορούμε να δούμε και δύο πειράματα σχετικά με την παραπάνω διαδικασία. Αρχικά, και στο σχήμα 23, πως υπάρχει ο μαγνήτης κοντά στον αισθητήρα Hall. Βλέπουμε πως το **LED** έχει ανάψει καθώς ανιχνεύει το μαγνητικό πεδίο. Επίσης, το αποτέλεσμα μπορούμε να το αντικρίσουμε και στο σχήμα 25. Ο αισθητήρας διαβάζει πεδίο με τιμή 9 mT. Έτσι, η συνθήκη του if loop ισχύει και δίνεται τάση στο λαμπάκι να ανάψει.



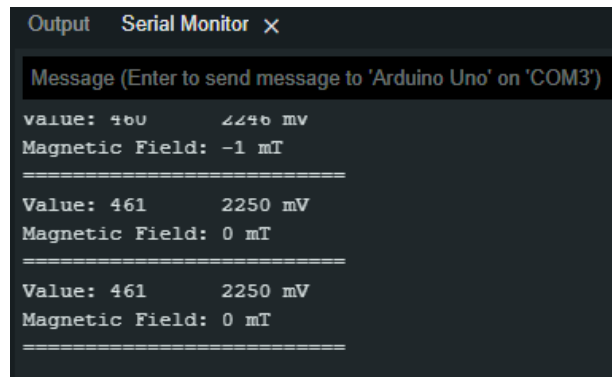
Σχήμα 25. Αποτέλεσμα μέτρησης απουσίας μαγνήτη από τον αισθητήρα.

Αντιθέτως, στην περίπτωση που ο μαγνήτης βρίσκεται σε απόσταση από τον αισθητήρα Hall, όπως παρατηρούμε και στο σχήμα 26, τότε δεν ισχύει η συνθήκη του if loop, καθώς το μαγνητικό πεδίο που ανιχνεύει είναι μικρότερο κατά απόλυτη τιμή 5 mT. Το αποτέλεσμα μπορούμε να το παρατηρήσουμε και στο σχήμα 27, όπου δεν υπάρχει η παρουσία μαγνήτη κοντά στο κύκλωμα.



Σχήμα 26. Κύκλωμα αισθητήρα με την απουσία μαγνήτη.





Σχήμα 27. Αποτέλεσμα μέτρηση με απουσία του μαγνήτη.

### 7.3 Ερώτημα 3

Σε αυτό το ερώτημα, τροποποιήθηκε αρχικά η χρόνος καθυστέρησης από τα 1000 στα 100 στην συνάρτηση `delay()`. Επίσης, τροποποιήθηκε και η λειτουργία του προγράμματος του προηγούμενου ερωτήματος, έτσι ώστε με την ανιχνευμένη τιμή του μαγνητικού πεδίου να έχει την σχετική φωτεινότητα και το **LED**. Δηλαδή, εάν ο μαγνήτης βρίσκεται πολύ κοντά στον αισθητήρα, τότε το **LED** να ανάβει πλήρως. Σε μία πιο απομακρυσμένη κοντινή θέση το **LED** να είναι αναμμένο με μικρότερη φωτεινότητα. Τέλος, στην περίπτωση που δεν ανιχνεύει μαγνητικό πεδίο τότε το **LED** δεν ανάβει καθόλου.

```

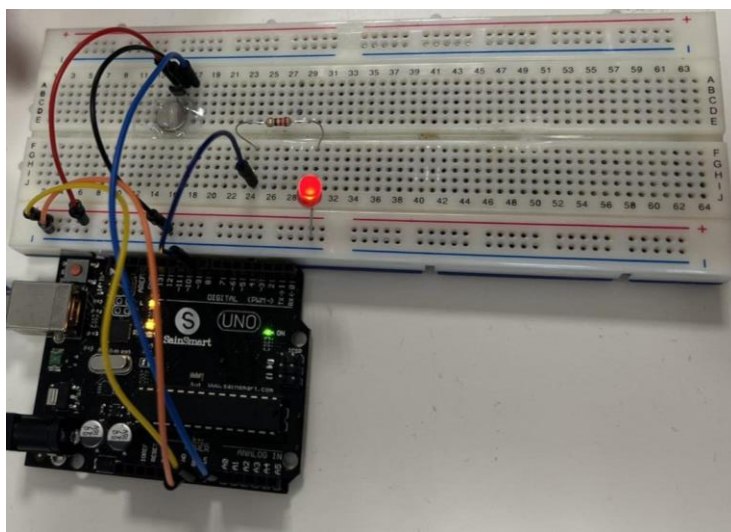
1  int pin = A0;           //Pin εισόδου αισθητήρα Hall
2  int vcc = 5000;        //Τάση τροφοδοσίας αισθητήρα Hall σε mV
3
4  int value;              //Μεταβλητή τιμής στο εύρος 0-1023
5  int voltage;            //Μεταβλητή τάσης στο εύρος 0-Vcc
6  int field;              //Μεταβλητή τιμής πεδίου σε mT
7  int offset = 254;      //Μεταβλητή τάσης offset σε mV
8  //int field_offset = -17; //Μεταβλητή τιμής πεδίου offset σε mT
9  int led = 9;
10 int ledValue;
11 void setup() {
12     Serial.begin(9600); //Εναρξη σειριακής επικοινωνίας
13     pinMode(led, OUTPUT);
14 }
15
16 void loop() {
17     value = analogRead(pin); //Ανάγνωση τιμής του αισθητήρα Hall
18     voltage = value * (vcc / 1024.0); //Μετατροπή σε τιμή τάσης (mV)
19     field = map(voltage+offset, 1000, 4000, -100, 100); //Υπολογισμός μαγνητικού πεδίου (mT)
20
21     ledValue=map(abs(field), 0, 100, 0, 255);
22     analogWrite(led, ledValue);
23
24
25     //Εμφάνιση αποτελεσμάτων
26     Serial.print("Value: ");
27     Serial.print(value);
28     Serial.print("\t");
29     Serial.print(voltage);
30     Serial.println(" mV");
31     Serial.print("Magnetic Field: ");
32     Serial.print(field);
33     Serial.println(" mT");
34     Serial.println("=====");
35     delay(100); //Λήψη νέας τιμής κάθε 100msec
36 }
37

```

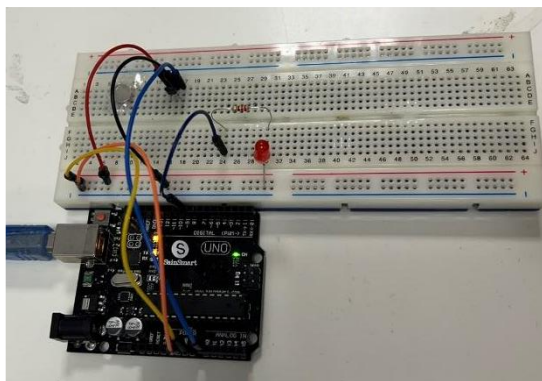
Σχήμα 28. Κώδικας τρίτου ερωτήματος.

Για να γίνει η παραπάνω διαδικασία, θα χρησιμοποιηθεί η συνάρτηση  $\text{map}(\text{abs}(\text{field}), 0, 100, 0, 255)$ . Ανάλογα με την τιμή εισόδου του μαγνητικού πεδίου θα αποτυπώνεται αντίστοιχα η φωτεινότητα του **LED**. Τέλος, γίνεται εγγραφή της μεταβλητής **ledValue** στο **pin led** με την χρήση της συνάρτησης `analogWrite()`. Έτσι, το **LED** θα δέχεται την κατάλληλη τιμή ανάλογα με την τιμή του μαγνητικού πεδίου.

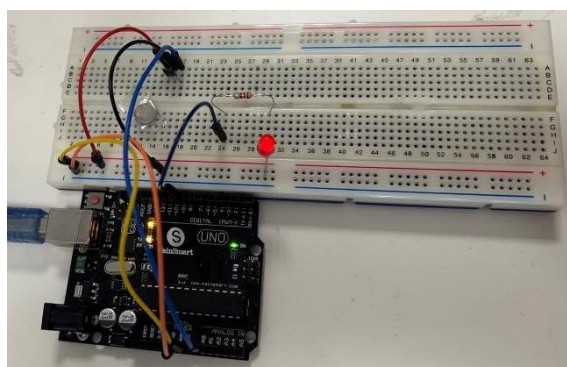
Στο σχήμα 29 παρατηρούμε την λειτουργία του **LED** όταν ο μαγνήτης βρίσκεται κοντά στον αισθητήρα. Βλέπουμε πως η φωτεινότητα του **LED** είναι στο 255. Με την απομάκρυνση του μαγνήτη από τον αισθητήρα η φωτεινότητα του **LED** μειώνεται. Στο σχήμα 30 παρατηρούμε πως με την απουσία του μαγνήτη από τον αισθητήρα δεν ανάβει το **LED**. Τέλος, στο σχήμα 31 παρατηρούμε πως το **LED** θα έχει χαμηλή φωτεινότητα. Αυτό οφείλεται στο γεγονός πως ο αισθητήρας ανιχνεύει μικρό μαγνητικό πεδίο. Έτσι, διαπιστώνουμε πως με την τροποποίηση του κώδικα, μπορούμε να ανιχνεύσουμε το μαγνητικό πεδίο με ένα **LED**.



Σχήμα 29. Ανίχνευση μεγάλης τιμής μαγνητικού πεδίου.



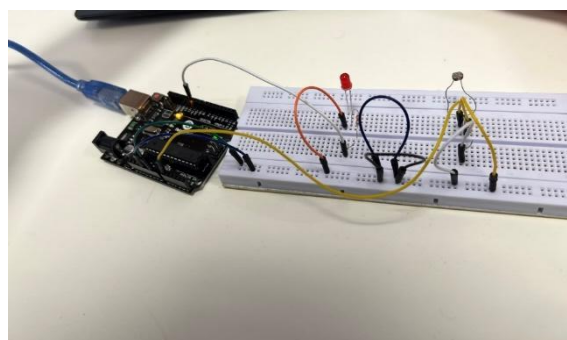
Σχήμα 30. Ανίχνευση μηδενικού μαγνητικού πεδίου.



Σχήμα 31. Ανίχνευση μαγνητικού πεδίου με μικρή τιμή.

## 8. Παράδειγμα – Αισθητήρας φωτεινότητας

Σε αυτό το παράδειγμα μας δίνεται ο κώδικας **8\_Light\_Sensor.ino**. Ο κώδικας αυτός είναι υπεύθυνος να διαβάζει είτε ψηφιακά είτε αναλογικά την έξοδο μιας φωτοαντίστασης και αναλόγως ανάβει ή σβήνει ένα **LED**. Η συνδεσμολογία φαίνεται στο σχήμα 32.

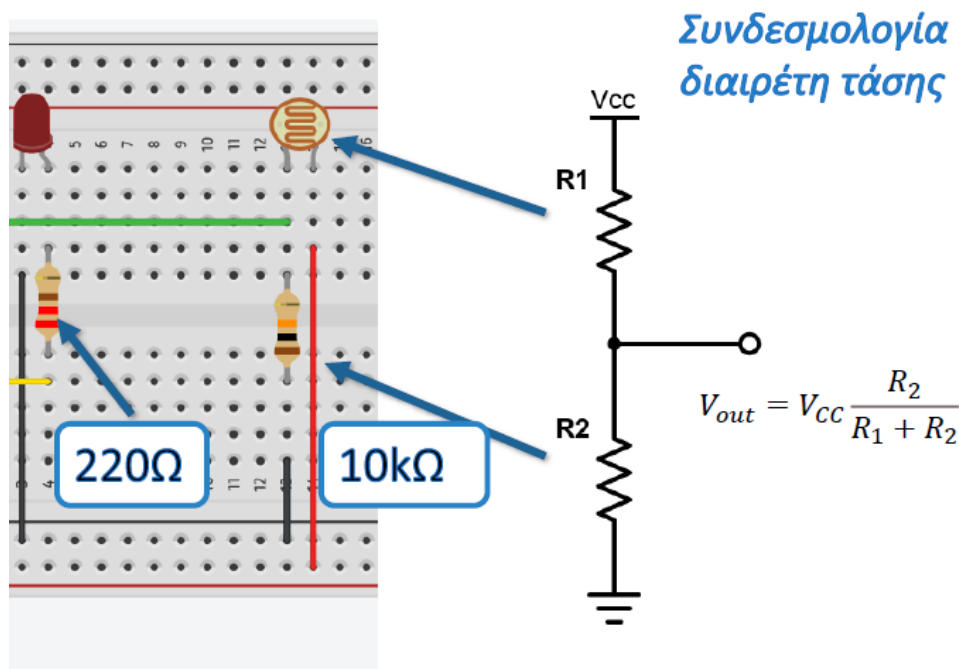


Σχήμα 32. Συνδεσμολογία κυκλώματος αισθητήρα φωτεινότητας.

Η αναλογική λειτουργία του προγράμματος επιτρέπει την φωτεινότητα του **LED** να ανάβει και να σβήνει ανάλογα με την τάση που διαβάζεται από την φωτοαντίσταση. Έτσι, θα προσομοιώνει όλες τις τιμές της τάσης. Η ψηφιακή λειτουργία του προγράμματος επιτρέπει στο **LED** να ανάβει και να σβήνει μια φορά. Όταν εκπέμπεται φως στην αντίσταση τότε το **LED** θα ανάβει. Όταν δεν εκπέμπεται φως στην αντίσταση τότε το λαμπάκι δεν θα ανάψει.

### 8.1 Ερώτημα 1

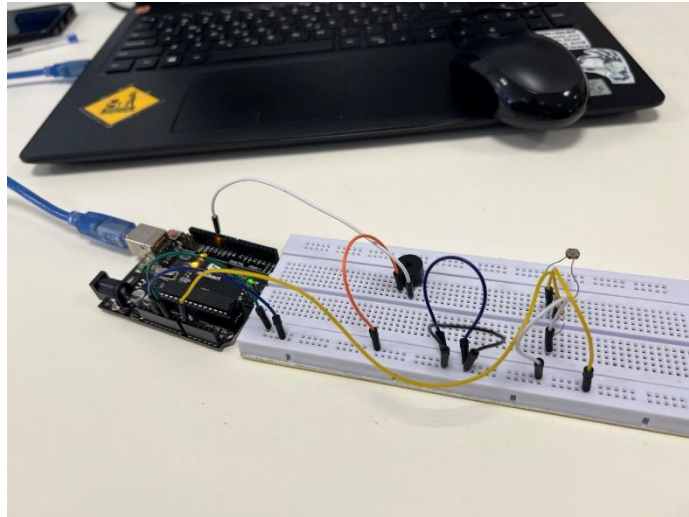
Στο παρών κύκλωμα υπάρχει ένας διαιρέτης τάσης. Η ύπαρξη του διαιρέτη τάσης επιτρέπει την μετατροπή της μέτρησης της φωτεινότητας από μεταβλητή αντίσταση σε μεταβλητή τάση, η οποία μπορεί να αναγνωριστεί από το Arduino.



Σχήμα 33. Κύκλωμα διαιρέτη τάσης φωτοτρανζίστορ.

### 8.2 Ερώτημα 2

Σε αυτό το ερώτημα είχαμε να τροποποιήσουμε τον δοθέν κώδικα και να τροποποιηθεί και το κύκλωμα της διάταξης, με την αφαίρεση του **LED** και την εισαγωγή ενός **buzzer**. Το νέο κύκλωμα παρουσιάζεται στο σχήμα 34.



Σχήμα 34. Τροποποιημένη συνδεσμολογία κυκλώματος με την χρήση buzzer.

Γνωρίζουμε πως το **buzzer** λειτουργεί σε συχνότητες από 100 έως 1000 Hz. Οπότε θα τροποποιήσουμε την συνάρτηση `map()` με το ανάλογο εύρος, τόσο στην συνάρτηση `analog()` όσο και στην συνάρτηση `digital()`. Έτσι, οι τιμές εξόδου που θα έχει η τιμή **analogValue** θα κυμαίνονται σε αυτό το εύρος. Στη συνέχεια με την εντολή `tone()` θα εισάγουμε αυτή την τιμή από το Arduino στην έξοδο 11 που είναι η είσοδος του buzzer. Αυτό θα γίνει με την εντολή `tone(speaker_pin, analogValue, 200)`. Έτσι, θα εφαρμόζεται η αντίστοιχη συχνότητα στο buzzer με διάρκεια 200 msec.

```
void analog() {  
    value = analogRead(A0);  
    Serial.print(value);  
    Serial.print("\t");  
    int analogValue = map(value, 512, 1023, 100, 1000);  
    // analogWrite(speaker_pin, analogValue);  
    tone(speaker_pin, analogValue, 200);  
    Serial.println(analogValue);  
}  
  
void digital() {  
    value = analogRead(A0);  
    Serial.print(value);  
    Serial.print("\t");  
    boolean digitalValue = map(value, 0, 600, HIGH, LOW);  
    // digitalWrite(speaker_pin, digitalValue);  
    tone(speaker_pin, digitalValue*500, 200);  
    Serial.println(digitalValue);  
}
```

Σχήμα 35. Κώδικας δεύτερου ερωτήματος.

Συνεπώς, με την τροποποίηση του κυκλώματος και του κώδικα θα έχουμε την αντίστοιχη λειτουργία. Εάν επιλεγθεί ο χαρακτήρας 'A' από το Serial Monitor, τότε ανάλογα με την εκπομπή του φωτός στην φωτοαντίσταση, θα δέχεται το buzzer όλο το εύρος συχνοτήτων της συνάρτησης *map()*. Στην περίπτωση που έχουμε εισάγει την τιμή 'D' από το Serial Monitor, τότε το buzzer θα διαβάζει μία μόνο συγκεκριμένη συχνότητα από την φωτοαντίσταση και θα παράγει ήχο. Ένα παράδειγμα λειτουργίας φαίνεται και στο σχήμα 36 που ακολουθεί.

```
Unknown command
827      0
825      0
823      0
Unknown command
835      161
838      162
840      163
Unknown command
```

Σχήμα 36. Αποτέλεσμα ανίχνευσης φωτός με την χρήση buzzer.