Φύλλο Εργαστηριακής Άσκησης:
Pulse Width Modulation (Διαμόρφωση Πλάτους Παλμού)

Ονοματεπώνυμο:

Ημερομηνία: Ομάδα:

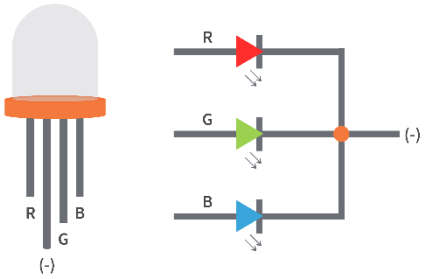
Σύνδεση με τα προηγούμενα

Με την ολοκλήρωση του προηγούμενου μαθήματος έχουμε δει πλέον πολλές πληροφορίες για τα Arduino, οπότε καλό είναι να τα έχουμε και κάπου μαζεμένα:

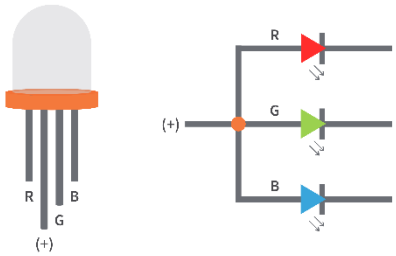
- Ακροδέκτες Arduino UNO
- «Arduino Projects Book»
- Ερωτηματολόγιο για έλεγχο της αντίληψης περί αναλογικών και ψηφιακών εισόδων και εξόδων (Με απαντήσεις).
- Εισαγωγικές σημειώσεις ηλεκτρικών κυκλωμάτων και ηλεκτρονικής για το μάθημα «Δρώμενα Πληκτρολογίου» | Συνοπτικό και χρήσιμο για την κατανόηση βασικών εννοιών.
- Κεντρική Σελίδα Arduino.
- Arduino IDE | Το βασικό πρόγραμμα για κώδικα Arduino.
- Scrarch for Arduino | S4A.
- ThinkerCad (by Autodesk).
- Επίσημα έγγραφα του Arduino | Έχει πληροφορίες για τα πάντα, αλλά μπορεί να γίνει πολύ τεχνικό.
- DI-STEM | Πρόγραμμα με εκπαιδευτικό υλικό για prject.
- Πώς λειτουργούν οι κινητήρες συνεχούς τάσης (DC motors).
- Πώς λειτουργούν τα transistor.
- Παράδειγμα πρότζεκτ | Ο δίσκος του Νεύτωνα με Arduino.
- Βιβλίο Εκπαιδευτικού από το DISTEM «Σπίτι μου, σπιτάκι μου».

Σήμερα, θα προσπαθήσουμε να μάθουμε να χειριζόμαστε την Διαμόρφωση Πλάτους Παλμού, μαζί με μερικές εντολές, οι οποίες θα φανούν χρήσιμες.

Γνωριμία με τα νέα «εξαρτήματα»



Σχήμα 2



Σχήμα 1

*Ένα **RGB LED** είναι ουσιαστικά 3 LED (κόκκινο, πράσινο και μπλε) τοποθετημένα στο ίδιο περίβλημα. Ανάλογα με το πώς είναι συνδεδεμένα εσωτερικά χωρίζονται σε δύο κατηγορίες:*

1) **Κοινής καθόδου**. Σε αυτό το σενάριο, οι **κάθοδοι** των 3 LED είναι **βραχυκυκλωμένες** μαζί και βγαίνουν σαν μία ακίδα. Αρκεί να συνδέσουμε χαμηλή τάση (για εμάς GRD) στην κοινή κάθοδο και να παρέχουμε θετική τάση σε κάποια από τις άλλες 3 ακίδες για να ανάψει το αντίστοιχο LED.
2) **Κοινής ανόδου.** Αντίστροφα από πριν, αυτή την φορά η υψηλή τάση είναι βραχυκυκλωμένη. Η κοινή ακίδα τροφοδοτείται με υψηλή τάση (για εμάς θα ήταν 5V) και οι άλλες ακίδες παίρνουν τάση χαμηλότερη για να λάμψουν τα αντίστοιχα LED.

Πολύ συχνά στα ηλεκτρονικά εξαρτήματα έχουμε είναι εξαρτήματα κοινής καθόδου, είτε εξαρτήματα κοινής ανόδου. Η επιλογή μεταξύ των δύο έχει να κάνει με το υπόλοιπο σύστημα στο οποίο συνδέονται. Για εμάς που τα συνδέουμε σε ένα Arduino, δεν παίζει κανέναν απολύτως ρόλο. Απλά θα πρέπει να βάλουμε σωστά την τάση.

Κοινά χαρακτηριστικά:

1) Σχεδόν πάντα η **μακρύτερη** ακίδα είναι η **κοινή** άνοδος ή κάθοδος.
2) Συνήθως για LED τέτοιου μεγέθους η **πτώση τάσης** στα led είναι:
    a. 3.4V για μπλε και πράσινο.
    b. 2.3V για το κόκκινο.
3) Συνήθως το ρεύμα λειτουργίας τους είναι 20mA ανά χρώμα.
4) Παρόλο που μοιάζει λογικό, είναι καλή πρακτική να συνδέσουμε 3 αντιστάσεις (μία για κάθε χρώμα), παρά μία στην κοινή έξοδο. Αυτό συμβαίνει λόγω των διαφορετικών πτώσεων τάσης που είδαμε.
5) Τα χρώματα συνήθως όταν κοιτάμε την δίοδο όπως φαίνεται στα παραπάνω σχήματα, πάνε **με την σειρά RGB** (κόκκινο, πράσινο, μπλε). Όμως αυτό πρέπει κάθε φορά να το ελέγχουμε. (Για παράδειγμα, στο thinkerCad το RGB LED που υπάρχει διαθέσιμο πάει RBG)
6) Αν τα συνδέσουμε ανάποδα **δεν καίγονται**, απλώς δεν θα περάσει ρεύμα και δεν θα ανάψουν.

Γνωριμία με τα νέες εντολές

Η εντολή analogWrite(pin, value**)**

Η εντολή **analogWrite(pin, value)** είναι ένας τρόπος με τον οποίο το Arduino μπορεί να παράγει ένα "ψευδο-αναλογικό" σήμα σε συγκεκριμένες ψηφιακές εξόδους. Στην πραγματικότητα, δημιουργεί ένα σήμα **Pulse Width Modulation (PWM)**, δηλαδή Διαμόρφωσης Εύρους Παλμού. Αυτό σημαίνει ότι η έξοδος ενεργοποιείται και απενεργοποιείται πολύ γρήγορα. Το ποσοστό του χρόνου που η έξοδος παραμένει ενεργοποιημένη (HIGH) σε σχέση με το συνολικό χρόνο ενός κύκλου ονομάζεται **κύκλος λειτουργίας (duty cycle)**. Τα μάτια μας ή οι κινητήρες αντιλαμβάνονται αυτόν τον κύκλο λειτουργίας ως μια μέση τιμή τάσης, επιτρέποντάς μας να ελέγξουμε π.χ. την φωτεινότητα ενός LED ή την ταχύτητα ενός κινητήρα.

Οι παράμετροι της εντολής είναι:

- **pin**: Ο αριθμός του pin στο οποίο θα εφαρμοστεί το σήμα PWM. Στο Arduino Uno, τα pins που υποστηρίζουν PWM είναι συνήθως τα 3, 5, 6, 9, 10, και 11 (συχνά φέρουν περισπωμένη «~» δίπλα στον αριθμό τους).
- **value**: Μια ακέραια τιμή από **0 έως 255**.
    - Μια τιμή 0 αντιστοιχεί σε κύκλο λειτουργίας 0% (η έξοδος είναι πάντα LOW/απενεργοποιημένη).
    - Μια τιμή 255 αντιστοιχεί σε κύκλο λειτουργίας 100% (η έξοδος είναι πάντα HIGH/πλήρως ενεργοποιημένη).
    - Ενδιάμεσες τιμές (π.χ. 127) αντιστοιχούν σε ενδιάμεσους κύκλους λειτουργίας (π.χ. περίπου 50%).

(ΠΡΟΣΟΧΗ: Δεν χρειάζεται να δηλώσετε το pin ως OUTPUT με pinMode() όταν χρησιμοποιείτε την analogWrite(); η εντολή το κάνει αυτόματα.)

Η συνάρτηση map(value, fromLow, fromHigh, toLow, toHigh**)**

Η συνάρτηση map() είναι ένα πολύ χρήσιμο εργαλείο στο Arduino που μας επιτρέπει να **αντιστοιχίσουμε** μια τιμή **από μια συγκεκριμένη αριθμητική κλίμακα σε μια άλλη**. Φανταστείτε ότι έχετε μια τιμή από έναν αισθητήρα (όπως την φωτοαντίσταση των προηγούμενων εργαστηρίων που δίνει τιμές από 650 έως 1023) και θέλετε να τη χρησιμοποιήσετε για να ελέγξετε κάτι άλλο που λειτουργεί σε διαφορετική κλίμακα (π.χ. τη φωτεινότητα ενός LED με analogWrite(), που δέχεται τιμές από 0 έως 255, ή τη γωνία ενός servo motor από 0 έως 180 μοίρες). Η map() κάνει αυτή τη μετατροπή για εμάς.

Οι παράμετροι της συνάρτησης είναι:

- **value**: Η αρχική τιμή που θέλουμε να αντιστοιχίσουμε.
- **fromLow**: Το κατώτερο όριο της αρχικής κλίμακας της τιμής value.
- **fromHigh**: Το ανώτερο όριο της αρχικής κλίμακας της τιμής value.
- **toLow**: Το κατώτερο όριο της επιθυμητής κλίμακας στην οποία θέλουμε να αντιστοιχίσουμε την τιμή.
- **toHigh**: Το ανώτερο όριο της επιθυμητής κλίμακας.

Η συνάρτηση επιστρέφει την αντιστοιχισμένη τιμή.

Προσέξτε:

- Η συνάρτηση map() χρησιμοποιεί **ακέραια αριθμητική.** Αυτό σημαίνει ότι τα δεκαδικά μέρη των υπολογισμών αποκόπτονται (δεν γίνεται στρογγυλοποίηση).
- Αν η αρχική value είναι **εκτός της κλίμακας fromLow - fromHigh,** η map() θα κάνει γραμμική επέκταση (extrapolation) της αντιστοίχισης. Αυτό μπορεί να οδηγήσει σε **τιμές εκτός της επιθυμητής κλίμακας toLow - toHigh** αν δεν το περιμένετε. Για παράδειγμα, **αν η value είναι μεγαλύτερη από το fromHigh, η επιστρεφόμενη τιμή θα είναι μεγαλύτερη από το toHigh.**
- Οι κλίμακες μπορούν να περιλαμβάνουν και **αρνητικούς αριθμούς.**

Δραστηριότητα 1 – Πόσο γρήγορη αλλαγή βλέπουν τα μάτια μας;

Στόχοι Δραστηριότητας:

Να παρατηρήσετε πώς η **ταχεία εναλλαγή** ενός LED **μεταξύ των καταστάσεων HIGH (αναμμένο) και LOW (σβηστό) μπορεί να επηρεάσει την αντιληπτή φωτεινότητά του**.

Να **πειραματιστείτε** με **διαφορετικούς χρόνους** παραμονής στις καταστάσεις HIGH και LOW, χρησιμοποιώντας χιλιοστά του δευτερολέπτου.

Να ανακαλύψετε το όριο συχνότητας πάνω από το οποίο το ανθρώπινο μάτι παύει να αντιλαμβάνεται το τρεμόπαιγμα.

Να κατανοήσετε την βασική αρχή πίσω από την τεχνική διαμόρφωσης εύρους παλμού (PWM) που θα εξερευνήσουμε αργότερα.

Απαιτούμενα Υλικά:

1 x Πλακέτα Arduino (π.χ., Uno)

1 x Breadboard

2 x LEDs ίδιου χρώματος (π.χ., κόκκινα)

2 x Αντιστάσεις 220Ω (ή κατάλληλες για τα LEDs σας)

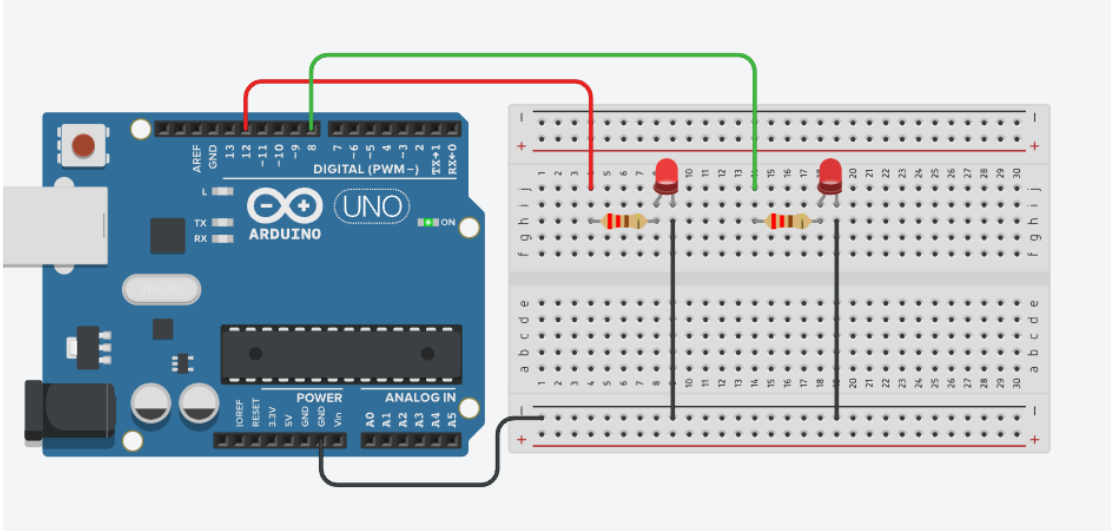
Καλώδια σύνδεσης

Συνδεσμολογία:

Συνδέστε το ένα LED (ας το ονομάσουμε LED_A) σε μια ψηφιακή ακίδα (π.χ., pin 12) μέσω μιας αντίστασης 220Ω. Αυτό το LED θα το χρησιμοποιήσουμε ως σημείο αναφοράς.

Συνδέστε το δεύτερο LED (ας το ονομάσουμε LED_B) σε μια άλλη ψηφιακή ακίδα (π.χ., pin 8) μέσω μιας αντίστασης 220Ω. Με αυτό το LED θα πειραματιστούμε.

Θυμηθείτε: Η μακρύτερη ακίδα του LED (άνοδος) συνδέεται προς την ακίδα του Arduino (μέσω της αντίστασης) και η κοντύτερη ακίδα (κάθοδος) συνδέεται στο GND.



Ο Κώδικας:

Θα χρησιμοποιήσουμε δύο μεταβλητές στην αρχή του κώδικά μας, τις timeON και timeOFF. Αυτές οι μεταβλητές θα καθορίζουν για πόσο χρόνο (τώρα σε χιλιοστά του δευτερολέπτου) το LED_B θα παραμένει αναμμένο και σβηστό αντίστοιχα σε κάθε κύκλο.

```
// Γενικές μεταβλητές για τους χρόνους ON και OFF (σε χιλιοστά
του δευτερολέπτου - ms)
// Αλλάξτε αυτές τις τιμές για να πειραματιστείτε! Σας
ενδιαφέρει:
// 1) Πόση είναι η Περίοδος (T)
// T = timeON + timeOFF σε ms
// 2) Πόσο ποσοτό του χρόνου είναι ανοιχτό το LED (κύκλος
εργασίας).
// Κύκλος Εργασίας = timeON / T x 100%
int timeON = 500;  // Αρχική τιμή: 500ms (μισό δευτερόλεπτο)
int timeOFF = 500; // Αρχική τιμή: 500ms (μισό δευτερόλεπτο)

// Ακίδες για τα LEDs
const int ledPinA = 12; // LED αναφοράς (σταθερά αναμμένο)
const int ledPinB = 8;  // LED πειραματισμού (αναβοσβήνει)

void setup() {
  pinMode(ledPinA, OUTPUT);
  pinMode(ledPinB, OUTPUT);

  digitalWrite(ledPinA, HIGH); // Ανάβουμε μόνιμα το LED αναφοράς
}

void loop() {
  digitalWrite(ledPinB, HIGH);   // Ανάβουμε το LED πειραματισμού
  delay(timeON);                 // Περιμένουμε για timeON
χιλιοστά του δευτερολέπτου

  digitalWrite(ledPinB, LOW);    // Σβήνουμε το LED πειραματισμού
  delay(timeOFF);                // Περιμένουμε για timeOFF
χιλιοστά του δευτερολέπτου
}
```

Συμπέρασμα

Τελικά πόσα millisecond αρκούν για να ξεγελάσουν τα μάτια μας;

Δραστηριότητα 2 – Έλεγχος Φωτεινότητας LED με analogWrite()

Στόχοι Δραστηριότητας:

- Να χρησιμοποιήσετε την εντολή analogWrite() για να ελέγξετε τη φωτεινότητα ενός LED (LED_B).
- Να παρατηρήσετε την ομαλή αλλαγή φωτεινότητας του LED_B και να τη συγκρίνετε με ένα άλλο LED (LED_A) που βρίσκεται σε πλήρη φωτεινότητα μέσω digitalWrite().
- Να συνδέσετε την εμπειρία από τη Δραστηριότητα 1 με τον τρόπο που η analogWrite() διαχειρίζεται τη φωτεινότητα.
-

Απαιτούμενα Υλικά: (Ίδια με τη Δραστηριότητα 1)

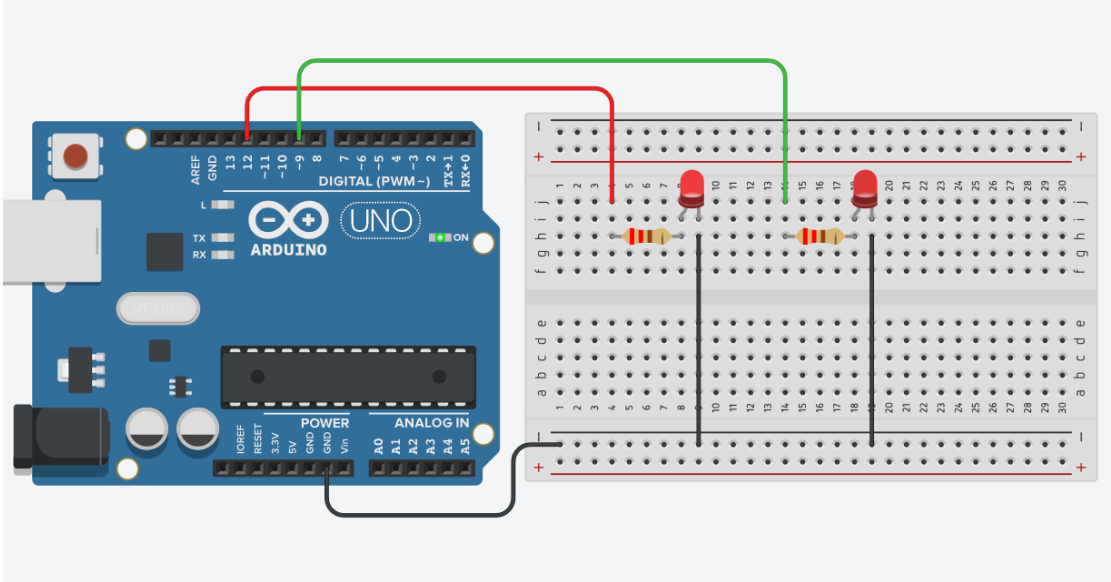
- 1 x Πλακέτα Arduino (π.χ., Uno)
- 1 x Breadboard
- 2 x LEDs ίδιου χρώματος (π.χ., κόκκινα)
- 2 x Αντιστάσεις 220Ω (ή κατάλληλες για τα LEDs σας)
- Καλώδια σύνδεσης

Συνδεσμολογία:

Η συνδεσμολογία είναι παρόμοια με τη Δραστηριότητα 1, με μια σημαντική αλλαγή για το LED που θα ελέγχεται με analogWrite():

1. LED_A (LED Αναφοράς): Συνδέστε το πρώτο LED (ας το ονομάσουμε LED_A) στην ψηφιακή ακίδα 12 του Arduino, μέσω μιας αντίστασης 220Ω. Αυτό το LED θα είναι το σημείο αναφοράς μας, συνεχώς αναμμένο σε πλήρη φωτεινότητα. (Ακριβώς όπως ήταν στη Δραστηριότητα 1).
2. LED_B (LED Πειραματισμού): Συνδέστε το δεύτερο LED (ας το ονομάσουμε LED_B) μέσω μιας αντίστασης 220Ω, στην ακίδα 9 του Arduino. Παρατηρήστε ότι η ακίδα 9 στο Arduino Uno φέρει το σύμβολο "~" δίπλα της. Αυτό υποδηλώνει ότι είναι μια PWM ακίδα, κατάλληλη για χρήση με την analogWrite().
    - ***Σημείωση:*** **Αν στη Δραστηριότητα 1 είχατε το LED πειραματισμού σε μια απλή ψηφιακή ακίδα όπως η 8, τώρα το μετακινούμε στην PWM ακίδα 9. Αυτή η αλλαγή είναι το κλειδί!**

3. Θυμηθείτε να συνδέσετε τις καθόδους (κοντύτερες ακίδες) και των δύο LEDs στο GND του Arduino.



Ο Κώδικας:

Ο παρακάτω κώδικας θα ανάψει το LED_A σε πλήρη φωτεινότητα και θα σας επιτρέψει να ελέγξετε τη φωτεινότητα του LED_B αλλάζοντας την τιμή της μεταβλητής brightnessValueLedB.

```
// Ακίδες για τα LEDs
const int ledPinA = 12; // LED αναφοράς (σταθερά αναμμένο) - Ψηφιακή ακίδα
const int ledPinB = 9;  // LED πειραματισμού με analogWrite() - **PWM Ακίδα (~9)**

// Μεταβλητή για την τιμή φωτεινότητας (0-255) για το ledPinB
// Αλλάξτε αυτή την τιμή στην αρχή του κώδικα για να πειραματιστείτε!
int brightnessValueLedB = 128; // Αρχική τιμή: περίπου μισή φωτεινότητα

void setup() {
  pinMode(ledPinA, OUTPUT);
  // Η pinMode(ledPinB, OUTPUT) δεν είναι αυστηρά απαραίτητη για την analogWrite(),
  // καθώς η συνάρτηση ρυθμίζει την ακίδα ως έξοδο PWM.

  // Ανάβουμε μόνιμα το LED αναφοράς (LED_A) σε πλήρη φωτεινότητα
  digitalWrite(ledPinA, HIGH);
}

void loop() {
  // Εφαρμογή της επιλεγμένης φωτεινότητας στο LED_B (στην PWM ακίδα)
  analogWrite(ledPinB, brightnessValueLedB);
}
```

Συμπέρασμα:

Ήταν πιο εύκολο να ελέγξετε τις διαφορετικές τιμές του LED με αυτήν την εντολή; Τι πιστεύεται πως ήταν καλύτερο; Συζητήστε.

Δραστηριότητα 3 – Ρύθμιση Έντασης LED με Ποτενσιόμετρο

Στόχοι Δραστηριότητας:

- Να διαβάσετε μια αναλογική τιμή από ένα ποτενσιόμετρο χρησιμοποιώντας την εντολή analogRead().
- Να χρησιμοποιήσετε τη συνάρτηση **map()** για να αντιστοιχίσετε την κλίμακα τιμών του ποτενσιόμετρου (0-1023) στην κλίμακα τιμών που απαιτεί η analogWrite() (0-255).
- Να ελέγξετε **δυναμικά** τη φωτεινότητα ενός LED περιστρέφοντας το ποτενσιόμετρο.
- Να εμπεδώσετε τη χρήση των αναλογικών εισόδων και εξόδων (PWM) του Arduino σε ένα ολοκληρωμένο κύκλωμα.

Απαιτούμενα Υλικά:

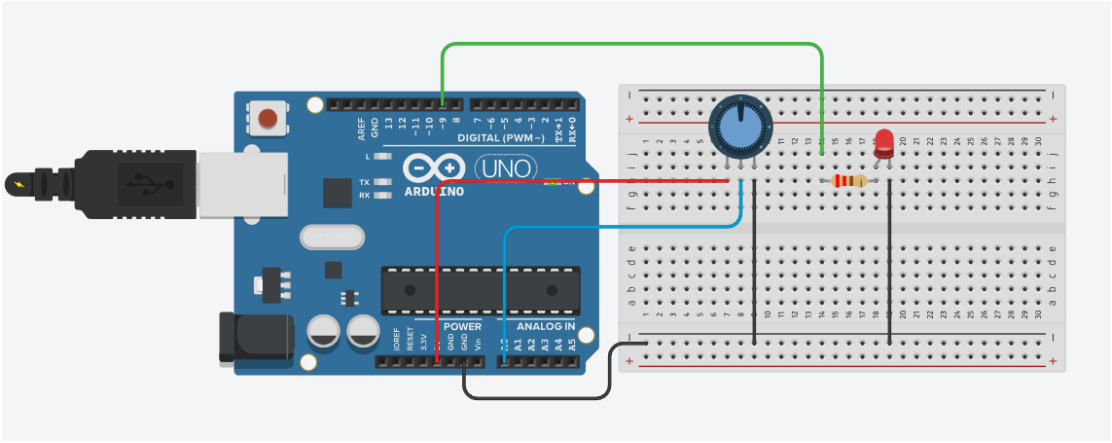
- 1 x Πλακέτα Arduino (π.χ., Uno)
- 1 x Breadboard
- 1 x LED (οποιουδήποτε χρώματος, π.χ., κόκκινο)
- 1 x Αντίσταση 220Ω (ή κατάλληλη για το LED σας)
- 1 x Ποτενσιόμετρο (π.χ., 10kΩ)
- Καλώδια σύνδεσης

Συνδεσμολογία:

1. **LED** (Ίδια με πριν)**:**
    - Συνδέστε την άνοδο του LED (μακρύτερη ακίδα) μέσω της αντίστασης 220Ω στην **PWM ακίδα 9** του Arduino (όπως και στην προηγούμενη δραστηριότητα).

- Συνδέστε την κάθοδο του LED (κοντύτερη ακίδα) στο **GND** του Arduino.

2. **Ποτενσιόμετρο:**
   - Το ποτενσιόμετρο έχει τρεις ακίδες.
   - Συνδέστε τη μία εξωτερική ακίδα του ποτενσιόμετρου στο **5V** του Arduino.
   - Συνδέστε την άλλη εξωτερική ακίδα του ποτενσιόμετρου στο **GND** του Arduino.
   - Συνδέστε τη μεσαία ακίδα του ποτενσιόμετρου σε μία από τις **αναλογικές εισόδους** του Arduino (π.χ., **A0**).



Εισαγωγή / Σύνδεση με Προηγούμενα:

Στην προηγούμενη δραστηριότητα, ελέγξαμε τη φωτεινότητα ενός LED αλλάζοντας χειροκίνητα μια τιμή στον κώδικα για την analogWrite(). Τώρα, θα κάνουμε αυτόν τον έλεγχο δυναμικό, χρησιμοποιώντας ένα ποτενσιόμετρο.

- Θα διαβάσουμε την τάση από τη μεσαία ακίδα του ποτενσιόμετρου με την analogRead(analogPin). Αυτή η συνάρτηση επιστρέφει μια τιμή από **0 έως 1023**, ανάλογα με τη θέση του δρομέα του ποτενσιόμετρου.
- Επειδή η analogWrite(ledPin, value) περιμένει μια τιμή από **0 έως 255**, θα χρειαστεί να "μεταφράσουμε" ή να "αντιστοιχίσουμε" την κλίμακα 0-1023 στην κλίμακα 0-255. Για αυτό, θα χρησιμοποιήσουμε τη συνάρτηση map() που έχετε ήδη γνωρίσει: mappedValue = map(originalValue, fromLow, fromHigh, toLow, toHigh);

Ο Κώδικας:

Συμπληρώστε τον παρακάτω κώδικα με τη λογική για την ανάγνωση του ποτενσιόμετρου, την αντιστοίχιση της τιμής και τον έλεγχο του LED.

```
// Ακίδες
const int ledPin = 9;          // PWM ακίδα για το LED (π.χ. ~9)
const int potPin = A0;         // Αναλογική ακίδα για το
ποτενσιόμετρο (π.χ. A0)

// Μεταβλητές
int potValue = 0;            // Για την αποθήκευση της τιμής από το
ποτενσιόμετρο
```

```
int brightnessValue = 0;   // Για την αποθήκευση της
αντιστοιχισμένης τιμής για το LED

void setup() {
  // Η pinMode για το ledPin δεν είναι αυστηρά απαραίτητη για την
analogWrite.
  // Η pinMode για το potPin (A0) δεν χρειάζεται, καθώς οι
αναλογικές ακίδες
  // είναι εξ ορισμού είσοδοι.
}

void loop() {
  // 1. Διαβάστε την τιμή από το ποτενσιόμετρο (0-1023)
  potValue = analogRead(potPin);

  // 2. Αντιστοιχίστε την τιμή του ποτενσιόμετρου (0-1023)
  //    στην κλίμακα φωτεινότητας του LED (0-255) χρησιμοποιώντας
τη map().
  //    Θυμηθείτε: map(value, fromLow, fromHigh, toLow, toHigh)
  brightnessValue = map(potValue, 0, 1023, 0, 255);

  // 3. Ελέγξτε τη φωτεινότητα του LED χρησιμοποιώντας την
αντιστοιχισμένη τιμή
  analogWrite(ledPin, brightnessValue);
 }
```

Δραστηριότητα 4 – Έλεγχος LED με Φωτοαντίσταση

Στόχοι Δραστηριότητας:

- Να συνδέσετε μια φωτοαντίσταση ως μέρος ενός κυκλώματος διαιρέτη τάσης για να λαμβάνετε μεταβαλλόμενες αναλογικές τιμές ανάλογα με το φωτισμό.
- Να πραγματοποιήσετε μια απλή "βαθμονόμηση" για να βρείτε το εύρος τιμών που δίνει η φωτοαντίσταση υπό διαφορετικές συνθήκες φωτισμού.
- Να χρησιμοποιήσετε τις τιμές αυτές και τη συνάρτηση map() για να ελέγξετε τη φωτεινότητα ενός LED, δημιουργώντας π.χ. ένα απλό "φως νυκτός".

Απαιτούμενα Υλικά:

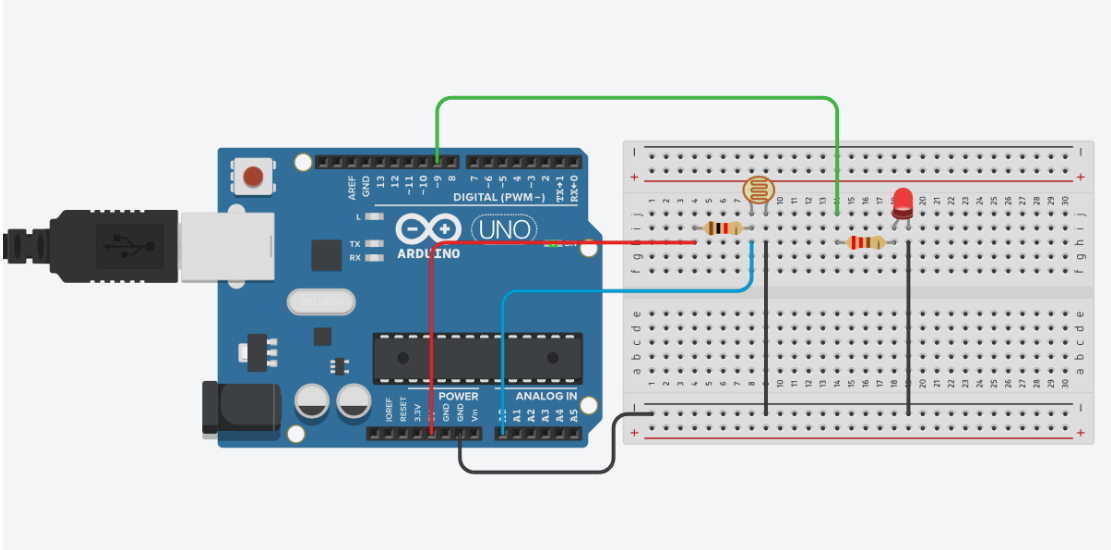
- 1 x Πλακέτα Arduino (π.χ., Uno)
- 1 x Breadboard
- 1 x LED (οποιουδήποτε χρώματος)
- 2 x Αντίσταση 220Ω (για το LED και για τον διαιρέτη τάσης με την φωτοαντίσταση. Ιδανικά θα θέλαμε μία 10κΩ, η οποία είναι πιο κοντά στις τιμές της φωτοαντίστασης, αλλά τότε δεν θα χρειαστεί να «παίξουμε» με την map(), οπότε κρατάμε τις ίδες.)
- 1 x Φωτοαντίσταση

- Καλώδια σύνδεσης

Συνδεσμολογία:

1. **LED:**
    - Συνδέστε την άνοδο του LED (μακρύτερη ακίδα) μέσω της αντίστασης 220Ω στην **PWM ακίδα 9** του Arduino.
    - Συνδέστε την κάθοδο του LED (κοντύτερη ακίδα) στο **GND** του Arduino.
2. **Φωτοαντίσταση (LDR) σε Διαιρέτη Τάσης:**
    - Συνδέστε το ένα άκρο της σταθερής αντίστασης 220Ω στο **5V** του Arduino.
    - Συνδέστε το άλλο άκρο της αντίστασης 220Ω σε μια αναλογική είσοδο, π.χ., στην **ακίδα A0**.
    - Συνδέστε το ένα "ποδαράκι" της φωτοαντίστασης στην ίδια γραμμή του breadboard με την ακίδα A0 (δηλαδή, στο ίδιο σημείο σύνδεσης της αντίστασης 10kΩ με το A0).
    - Συνδέστε το άλλο "ποδαράκι" της φωτοαντίστασης στο **GND** του Arduino.

Διάταξη Διαιρέτη Τάσης: 5V --- [Αντίσταση 220Ω] --- A0 (Σημείο Μέτρησης) --- [Φωτοαντίσταση] --- GND



Εισαγωγή / Σκεπτικό:

Μια **φωτοαντίσταση (LDR - Light Dependent Resistor)** είναι ένας ειδικός τύπος αντίστασης της οποίας η ηλεκτρική αντίσταση αλλάζει ανάλογα με την ποσότητα του φωτός που πέφτει πάνω της. Συνήθως:

- Σε έντονο φως, η αντίστασή της μειώνεται.
- Στο σκοτάδι, η αντίστασή της αυξάνεται σημαντικά.

Για να μπορέσουμε να "διαβάσουμε" αυτή την αλλαγή αντίστασης με το Arduino (το οποίο **μετράει τάσεις**, όχι απευθείας αντιστάσεις), τη συνδέουμε σε ένα κύκλωμα **διαιρέτη τάσης**.

Στη διάταξη που φτιάξαμε, η τάση στην ακίδα A0 θα αλλάζει καθώς αλλάζει η αντίσταση της LDR σε σχέση με τη σταθερή αντίσταση των 220Ω:

- **Περισσότερο φως** ⟹ Μικρότερη αντίσταση LDR ⟹ Η τάση στο A0 **πλησιάζει** προς το **GND** (χαμηλότερη τιμή στο analogRead()).
- **Λιγότερο φως (σκοτάδι)** ⟹ Μεγαλύτερη αντίσταση LDR ⟹ Η τάση στο A0 **πλησιάζει** προς τα 5V (υψηλότερη τιμή στο analogRead().
-

Πριν ελέγξουμε το LED, πρέπει να δούμε τι τιμές μας δίνει η φωτοαντίσταση στο περιβάλλον μας.

Βήμα 0: Κατανόηση και "Βαθμονόμηση" της Φωτοαντίστασης

Σε αυτό το βήμα, θα διαβάσουμε τις τιμές από τη φωτοαντίσταση και θα τις εμφανίσουμε στο Serial Monitor για να καταλάβουμε το εύρος λειτουργίας της στο δικό σας περιβάλλον φωτισμού.

Ο Κώδικας για το Βήμα 0:

```
const int ldrPin = A0; // Η αναλογική ακίδα όπου είναι
συνδεδεμένη η LDR
int ldrValue = 0;      // Μεταβλητή για την τιμή της LDR

void setup() {
  Serial.begin(9600);  // Έναρξη σειριακής επικοινωνίας
  // Δεν χρειάζεται pinMode για την αναλογική είσοδο A0
}

void loop() {
  ldrValue = analogRead(ldrPin); // Διαβάζουμε την τιμή από την
LDR

  Serial.print("Τιμή Φωτοαντίστασης (0-1023): ");
  Serial.println(ldrValue);

  delay(200); // Καθυστέρηση για να προλαβαίνουμε να διαβάζουμε
τις τιμές
}
```

Πειραματισμός Βήματος 0:

1. Ανεβάστε τον παραπάνω κώδικα στο Arduino σας.
2. Ανοίξτε το Serial Monitor (Tools > Serial Monitor, βεβαιωθείτε ότι το baud rate είναι 9600).
3. Παρατηρήστε τις τιμές που εκτυπώνονται:
    - Καλύψτε εντελώς τη φωτοαντίσταση με το δάχτυλό σας ή ένα αδιαφανές αντικείμενο (προσομοιώνοντας σκοτάδι). Σημειώστε την τιμή που βλέπετε. Αυτή θα είναι περίπου η **μέγιστη τιμή σας στο σκοτάδι** (ldrValue_dark).

- Εκθέστε τη φωτοαντίσταση σε έντονο φως (π.χ., Φακός κινητού). Σημειώστε την τιμή που βλέπετε. Αυτή θα είναι περίπου η **ελάχιστη τιμή σας στο φως** (ldrValue_light).

4. Καταγράψτε αυτές τις δύο τιμές. Για παράδειγμα:
   - ldrValue_dark (π.χ., περίπου 600-800, ανάλογα την LDR και το πόσο καλά την καλύψατε)
   - ldrValue_light (π.χ., περίπου 50-200, ανάλογα την LDR και την ένταση του φωτός)
   - ***Οι δικές σας τιμές μπορεί να διαφέρουν!***

Αυτές οι τιμές που σημειώσατε είναι σημαντικές για το επόμενο βήμα, όπου θα τις χρησιμοποιήσουμε στη συνάρτηση map().

Βήμα 1: Έλεγχος του LED με βάση τις τιμές της Φωτοαντίστασης

Τώρα θα χρησιμοποιήσουμε τις τιμές που βρήκαμε για να ελέγξουμε τη φωτεινότητα του LED. Θέλουμε το LED να λειτουργεί σαν "φως νυκτός": να ανάβει περισσότερο όταν υπάρχει σκοτάδι και λιγότερο (ή καθόλου) όταν υπάρχει φως.

Ο Κώδικας για το Βήμα 1: Προσαρμόστε τον παρακάτω κώδικα, αντικαθιστώντας τις τιμές YOUR_LDR_VALUE_IN_LIGHT και YOUR_LDR_VALUE_IN_DARK με αυτές που σημειώσατε στο Βήμα 0.

```
// Ακίδες
const int ledPin = 9;     // PWM ακίδα για το LED
const int ldrPin = A0;    // Αναλογική ακίδα για την LDR

// Μεταβλητές
int ldrValue = 0;         // Για την τιμή της LDR
int brightnessValue = 0; // Για την αντιστοιχισμένη τιμή
φωτεινότητας του LED

// *** ΠΡΟΣΑΡΜΟΣΤΕ ΑΥΤΕΣ ΤΙΣ ΤΙΜΕΣ ΜΕ ΒΑΣΗ ΤΟ ΒΗΜΑ 0! ***
int ldrValue_light_observed = 0; // Αντικαταστήστε με τη δική σας
τιμή στο φως
int ldrValue_dark_observed = 1023;  // Αντικαταστήστε με τη δική
σας τιμή στο σκοτάδι

void setup() {
}

void loop() {
  // 1. Διαβάστε την τρέχουσα τιμή από τη φωτοαντίσταση
  ldrValue = analogRead(ldrPin);

  // 2. Αντιστοιχίστε την τιμή της LDR στην κλίμακα φωτεινότητας
του LED.
  // Θέλουμε το LED να είναι πιο φωτεινό στο σκοτάδι (υψηλή τιμή
LDR στη διάταξή μας)
  // και πιο αχνό στο φως (χαμηλή τιμή LDR).
  // Άρα, αντιστοιχίζουμε:
  // ldrValue_light_observed -> 0 (LED σβηστό)
  // ldrValue_dark_observed  -> 255 (LED πλήρως αναμμένο)
```

```
  brightnessValue = map(ldrValue, ldrValue_light_observed,
ldrValue_dark_observed, 0, 255);

  // 3. Ελέγξτε τη φωτεινότητα του LED
  analogWrite(ledPin, brightnessValue);
}
```

Δραστηριότητα 5 – Έλεγχος RGB LED (Κοινής Καθόδου) και Δημιουργία Χρωμάτων

Στόχοι Δραστηριότητας:

- Να κατανοήσετε τη δομή και τη βασική λειτουργία ενός RGB LED τύπου κοινής καθόδου.
- Να ελέγξετε τα τρία βασικά χρώματα (Κόκκινο, Πράσινο, Μπλε) ενός RGB LED, χρησιμοποιώντας την εντολή analogWrite() σε τρεις διαφορετικές PWM ακίδες.
- Να πειραματιστείτε με τη δημιουργία διαφόρων χρωμάτων, ορίζοντας διαφορετικές εντάσεις (0-255) για κάθε ένα από τα τρία βασικά χρώματα.

Απαιτούμενα Υλικά:

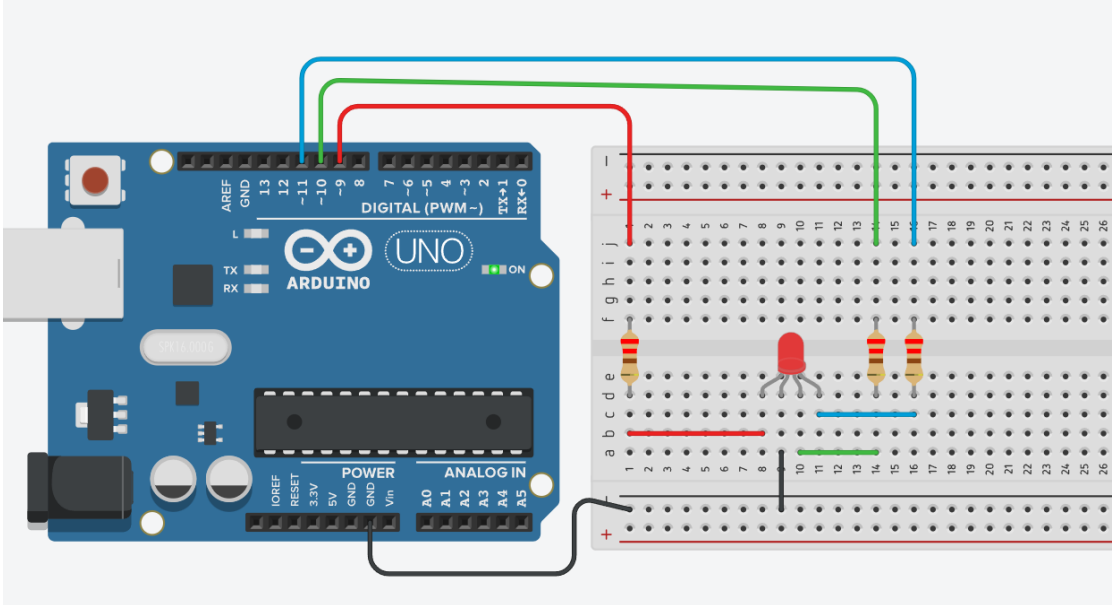
- 1 x Πλακέτα Arduino (π.χ., Uno)
- 1 x Breadboard
- 1 x RGB LED 4 ακίδων, τύπου **Κοινής Καθόδου (Common Cathode)**
- 3 x Αντιστάσεις 220Ω (Σημείωση: Η ακριβής τιμή της αντίστασης μπορεί να διαφέρει ανάλογα με τις προδιαγραφές του RGB LED σας για βέλτιστη φωτεινότητα και ασφάλεια. Οι αντιστάσεις 220Ω έως 330Ω είναι συνήθως μια καλή αρχή για λειτουργία με 5V.)
- Καλώδια σύνδεσης

Συνδεσμολογία:

Τα RGB LEDs έχουν συνήθως 4 ακίδες:

- Μία για το Κόκκινο (Red - R)
- Μία για το Πράσινο (Green - G)
- Μία για το Μπλε (Blue - B)
- Μία **κοινή ακίδα**. Στο LED κοινής καθόδου, αυτή είναι η μακρύτερη ακίδα και συνδέεται στο GND.

1. Τοποθετήστε το RGB LED στο breadboard.
2. Εντοπίστε τη μακρύτερη ακίδα του RGB LED (αυτή είναι η κοινή κάθοδος) και συνδέστε την στο **GND** του Arduino.
3. Για κάθε μία από τις υπόλοιπες τρεις ακίδες (R, G, B):
    - Συνδέστε μια αντίσταση 220Ω στη σειρά με την ακίδα χρώματος.
    - Συνδέστε την άλλη άκρη της αντίστασης σε μια **PWM ακίδα** του Arduino. Προτείνεται να χρησιμοποιήσετε:
        - Ακίδα Κόκκινου (R) → PWM ακίδα **~9**
        - Ακίδα Πράσινου (G) → PWM ακίδα **~10**
        - Ακίδα Μπλε (B) → PWM ακίδα **~11**



!Προσοχή! Στο THinkerCad δεν θα έπρεπε να συνδεθεί έτσι η φωτοδίοδος.

Εισαγωγή / Σκεπτικό:

Ένα RGB LED στην πραγματικότητα περιέχει τρία μικρότερα LEDs (ένα κόκκινο, ένα πράσινο και ένα μπλε) μέσα στο ίδιο περίβλημα. Στο LED **κοινής καθόδου**, όλες οι κάθοδοι αυτών των τριών εσωτερικών LEDs είναι συνδεδεμένες εσωτερικά και εξέρχονται ως μία κοινή ακίδα, την οποία συνδέουμε στο GND. Οι τρεις άλλες ακίδες είναι οι άνοδοι για κάθε χρώμα αντίστοιχα.

Για να ανάψουμε ένα συγκεκριμένο χρώμα (ή να ρυθμίσουμε την έντασή του), εφαρμόζουμε θετική τάση (ή ένα PWM σήμα) στην αντίστοιχη ακίδα ανόδου (R, G, ή B). Ελέγχοντας την ένταση κάθε ενός από τα τρία βασικά χρώματα με την analogWrite() (τιμές 0-255), μπορούμε να τα αναμείξουμε (προσθετική ανάμειξη χρωμάτων) για να δημιουργήσουμε ένα ευρύ φάσμα άλλων χρωμάτων, ακόμα και λευκό.

Ο Κώδικας:

Ο παρακάτω κώδικας ορίζει κάποιες βασικές τιμές για διάφορα χρώματα και στη συνέχεια τα εμφανίζει κυκλικά στο RGB LED.

```
// Ακίδες PWM για το RGB LED
const int redPin = 9;     // Ακίδα για το Κόκκινο στοιχείο
const int greenPin = 10; // Ακίδα για το Πράσινο στοιχείο
```

```
const int bluePin = 11;  // Ακίδα για το Μπλε στοιχείο

// Ορισμός μερικών βασικών χρωμάτων (Τιμές R, G, B από 0 έως 255)

// Χρώμα 1: Καθαρό Κόκκινο
const int valR1 = 255; const int valG1 = 0;   const int valB1 =
0;
// Χρώμα 2: Καθαρό Πράσινο
const int valR2 = 0;   const int valG2 = 255; const int valB2 =
0;
// Χρώμα 3: Καθαρό Μπλε
const int valR3 = 0;   const int valG3 = 0;   const int valB3 =
255;
// Χρώμα 4: Κίτρινο (Κόκκινο + Πράσινο)
const int valR4 = 255; const int valG4 = 255; const int valB4 =
0;
// Χρώμα 5: Μωβ (Κόκκινο + Μπλε)
const int valR5 = 255; const int valG5 = 0;   const int valB5 =
255;
// Χρώμα 6: Τιρκουάζ;; (Πράσινο + Μπλε)
const int valR6 = 0;   const int valG6 = 255; const int valB6 =
255;
// Χρώμα 7: Λευκό (Κόκκινο + Πράσινο + Μπλε)
const int valR7 = 255; const int valG7 = 255; const int valB7 =
255;

void setup() {
  // Ορισμός των ακίδων ως εξόδους
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  // Εμφάνιση Χρώματος 1 (Κόκκινο)
  analogWrite(redPin, valR1);
  analogWrite(greenPin, valG1);
  analogWrite(bluePin, valB1);
  delay(1500); // Αναμονή 1.5 δευτερόλεπτο

  // Εμφάνιση Χρώματος 2 (Πράσινο)
  analogWrite(redPin, valR2);
  analogWrite(greenPin, valG2);
  analogWrite(bluePin, valB2);
  delay(1500);

  // Εμφάνιση Χρώματος 3 (Μπλε)
  analogWrite(redPin, valR3);
  analogWrite(greenPin, valG3);
  analogWrite(bluePin, valB3);
  delay(1500);

  // Εμφάνιση Χρώματος 4 (Κίτρινο)
  analogWrite(redPin, valR4);
  analogWrite(greenPin, valG4);
  analogWrite(bluePin, valB4);
  delay(1500);

  // Εμφάνιση Χρώματος 5 (Μωβ)
```

```
  analogWrite(redPin, valR5);
  analogWrite(greenPin, valG5);
  analogWrite(bluePin, valB5);
  delay(1500);

  // Εμφάνιση Χρώματος 6 (Τιρκουάζ)
  analogWrite(redPin, valR6);
  analogWrite(greenPin, valG6);
  analogWrite(bluePin, valB6);
  delay(1500);

  // Εμφάνιση Χρώματος 7 (Λευκό)
  analogWrite(redPin, valR7);
  analogWrite(greenPin, valG7);
  analogWrite(bluePin, valB7);
  delay(1500);
}
```

Όπως παρατηρείτε, τα χρώματα δεν είναι ακριβώς αυτά που θα θέλαμε. Παίξτε με τις RGB τιμές μέχρι να δείχνει τα χρώματα που θα θέλατε εσείς. Αν νιώθετε «Θαρραλέοι» , μπορείτε να πάτε και στην Μπόνους δραστηριότητα!

Μπόνους Εντολές και δραστηριότητες

Η συνάρτηση Serial.read()

Η συνάρτηση Serial.read() χρησιμοποιείται για την **ανάγνωση εισερχόμενων δεδομένων** από τη **σειριακή θύρα**. Κάθε φορά που **καλείται**, διαβάζει έναν χαρακτήρα (ένα byte) από την προσωρινή μνήμη (buffer) των εισερχόμενων σειριακών δεδομένων. Αυτό είναι ιδιαίτερα χρήσιμο όταν θέλουμε το Arduino να λαμβάνει εντολές ή δεδομένα από τον υπολογιστή (μέσω του Serial Monitor) **ή από άλλη συσκευή που επικοινωνεί σειριακά**.

Πώς λειτουργεί:

Επιστρέφει τον πρώτο διαθέσιμο χαρακτήρα (byte) από την ουρά των εισερχόμενων σειριακών δεδομένων. Η τιμή που επιστρέφεται είναι τύπου int, αλλά συνήθως την μετατρέπουμε (cast) σε char όταν περιμένουμε χαρακτήρες.

Σημαντικό: Εάν δεν υπάρχουν διαθέσιμα δεδομένα στην ουρά για ανάγνωση, η Serial.read() επιστρέφει την τιμή -1 (ως int).

Για να χρησιμοποιήσουμε σωστά τη Serial.read(), σχεδόν **πάντα τη συνδυάζουμε πρώτα** με έναν **έλεγχο μέσω της Serial.available().** Η Serial.available() επιστρέφει τον αριθμό των χαρακτήρων (bytes) που έχουν φτάσει και περιμένουν στην ουρά για ανάγνωση. Επομένως, μια τυπική προσέγγιση είναι να καλούμε τη Serial.read() μόνο αφού βεβαιωθούμε ότι Serial.available() > 0.

Π.χ.

```
1. if (Serial.available() > 0) { // Αν υπάρχουν, τότε διαβάζουμε
τον επόμενο χαρακτήρα
2. char incomingChar = Serial.read();
3. Serial.print("Ελήφθη ο χαρακτήρας: ");
4. Serial.println(incomingChar);
5. } // Κλείνει το if
6.
```

Προσέξτε:

- Η Serial.read() επιστρέφει έναν **int** ώστε να μπορεί να επιστρέψει την ειδική τιμή -1 όταν δεν υπάρχουν δεδομένα. Όταν είστε σίγουροι ότι υπάρχουν δεδομένα (μετά από έλεγχο με Serial.available()), μπορείτε να **αντιστοιχίσετε** το αποτέλεσμα σε μια μεταβλητή **char**.
- Η συνάρτηση διαβάζει ένα byte (έναν χαρακτήρα) κάθε φορά. Αν στείλετε τη λέξη "**STEM**", το Arduino θα διαβάσει '**S**' στην **πρώτη κλήση** της Serial.read() (αν είναι διαθέσιμο), μετά '**T**' **στην επόμενη**, κ.ο.κ.
- Η σειριακή επικοινωνία χρειάζεται χρόνο. Η **εντολή Serial.begin(baudrate) πρέπει οπωσδήποτε να έχει κληθεί μέσα στη συνάρτηση setup().**

- Ο σειριακός buffer (η προσωρινή μνήμη όπου αποθηκεύονται τα εισερχόμενα bytes μέχρι να τα διαβάσετε) έχει **περιορισμένο μέγεθος** (συνήθως 64 bytes για τα περισσότερα Arduinos). Αν δεδομένα φτάνουν πιο γρήγορα από ό,τι τα διαβάζετε, ο buffer μπορεί να γεμίσει και νέα δεδομένα **μπορεί να χαθούν**.

Η εντολή switch ... case

Η εντολή switch ... case είναι μια **δομή ελέγχου ροής** που μας επιτρέπει να **επιλέξουμε ένα από πολλά διαφορετικά μονοπάτια εκτέλεσης κώδικα**, βασιζόμενοι στην τιμή μιας μεταβλητής. Είναι ιδιαίτερα χρήσιμη όταν θέλουμε να συγκρίνουμε μια μεταβλητή με μια σειρά από συγκεκριμένες, διακριτές τιμές, και αποτελεί μια πιο καθαρή εναλλακτική σε μια μακριά αλυσίδα από εντολές if...else if...else.

Σύνταξη και Βασικά Μέρη:

```
 1. switch (variable) {
 2.   case value1:
 3.     // Κώδικας που εκτελείται αν η variable == value1
 4.     break; // Πολύ σημαντικό!
 5.   case value2:
 6.     // Κώδικας που εκτελείται αν η variable == value2
 7.     break; // Πολύ σημαντικό!
 8.   // ... μπορείτε να έχετε περισσότερα cases
 9.   default: // Προαιρετικό
10.     // Κώδικας που εκτελείται αν η variable δεν ταιριάζει με
κανένα case
11.     break; // Καλό είναι να υπάρχει κι εδώ
12. }
13.
```

- **switch** (variable): Εδώ τοποθετούμε τη μεταβλητή (πρέπει να είναι ακέραιου τύπου, όπως int, char, byte) της οποίας την τιμή θέλουμε να ελέγξουμε.
- **case valueX**:: Κάθε case ακολουθείται από μια σταθερή τιμή (π.χ., 1, 'A', μια const int). Αν η τιμή της variable ταιριάζει με αυτή τη valueX, τότε εκτελείται ο κώδικας που ακολουθεί την άνω και κάτω τελεία.
- **break**;: Αυτή η εντολή είναι **κρίσιμη**. Όταν συναντηθεί, η εκτέλεση βγαίνει έξω από ολόκληρη τη δομή switch. **Αν παραλειφθεί, η εκτέλεση θα "πέσει" (fall-through) και θα συνεχίσει στον κώδικα του επόμενου case, ανεξάρτητα από την τιμή του!**
- **default**:: Αυτό το τμήμα είναι προαιρετικό. Ο κώδικας που περιέχει εκτελείται αν η τιμή της variable δεν ταιριάζει με καμία από τις τιμές των case.
-

Πώς λειτουργεί:

1. Η τιμή της variable μέσα στην παρένθεση του switch υπολογίζεται μία φορά.
2. Αυτή η τιμή **συγκρίνεται διαδοχικά** με τις τιμές που ορίζονται σε κάθε case.
3. Μόλις βρεθεί μια **αντιστοιχία**, **εκτελείται** ο **κώδικας** που σχετίζεται με αυτό το case.
4. Η εκτέλεση συνεχίζεται **μέχρι** να συναντήσει μια **εντολή break** (οπότε και βγαίνει από το switch) ή μέχρι το τέλος του μπλοκ switch.
5. **Αν δεν βρεθεί καμία αντιστοιχία** και υπάρχει **default τμήμα**, εκτελείται ο κώδικας του default.

Δραστηριότητα 6 – Διαδραστικός Έλεγχος RGB LED με Εντολές από το Serial Monitor

Στόχοι Δραστηριότητας:

- Να μπορείτε να λαμβάνετε και να επεξεργάζεστε είσοδο χαρακτήρων από το Serial Monitor του Arduino IDE, χρησιμοποιώντας τις συναρτήσεις Serial.available() και Serial.read().
- Να εφαρμόσετε τη δομή ελέγχου switch...case για να εκτελέσετε διαφορετικές ενέργειες (στην περίπτωσή μας, την αλλαγή του χρώματος ενός RGB LED) ανάλογα με τον χαρακτήρα που λαμβάνεται.
- Να συνδυάσετε τις γνώσεις σας για τον έλεγχο ενός RGB LED (όπως στη Δραστηριότητα 5), τη σειριακή επικοινωνία και τις δομές ελέγχου για να δημιουργήσετε ένα απλό, διαδραστικό σύστημα.

Απαιτούμενα Υλικά: (Ίδια με τη Δραστηριότητα 5)

- 1 x Πλακέτα Arduino (π.χ., Uno)
- 1 x Breadboard
- 1 x RGB LED 4 ακίδων, τύπου **Κοινής Καθόδου (Common Cathode)**
- 3 x Αντιστάσεις 220Ω (Σημείωση: Η ακριβής τιμή της αντίστασης μπορεί να διαφέρει ανάλογα με τις προδιαγραφές του RGB LED σας για βέλτιστη φωτεινότητα και ασφάλεια. Οι αντιστάσεις 220Ω έως 330Ω είναι συνήθως μια καλή αρχή για λειτουργία με 5V.)
- Καλώδια σύνδεσης

Συνδεσμολογία: (Ίδια με τη Δραστηριότητα 5) Τα RGB LEDs έχουν συνήθως 4 ακίδες:

- Μία για το Κόκκινο (Red - R)
- Μία για το Πράσινο (Green - G)
- Μία για το Μπλε (Blue - B)
- Μία κοινή ακίδα. Στο LED κοινής καθόδου, αυτή είναι η μακρύτερη ακίδα και συνδέεται στο GND.

1. Τοποθετήστε το RGB LED στο breadboard.
2. Εντοπίστε τη μακρύτερη ακίδα του RGB LED (αυτή είναι η κοινή κάθοδος) και συνδέστε την στο **GND** του Arduino.
3. Για κάθε μία από τις υπόλοιπες τρεις ακίδες (R, G, B):
    - Συνδέστε μια αντίσταση 220Ω στη σειρά με την ακίδα χρώματος.
    - Συνδέστε την άλλη άκρη της αντίστασης σε μια **PWM ακίδα** του Arduino. Προτείνεται να χρησιμοποιήσετε:
        - Ακίδα Κόκκινου (R) → PWM ακίδα **~9**
        - Ακίδα Πράσινου (G) → PWM ακίδα **~10**

- Ακίδα Μπλε (B) → PWM ακίδα **~11**

Εισαγωγή / Σκεπτικό:

Στην προηγούμενη δραστηριότητα, μάθαμε πώς να ελέγχουμε ένα RGB LED για να παράγει διάφορα προκαθορισμένα χρώματα. Τώρα, θα κάνουμε αυτόν τον έλεγχο **διαδραστικό**! Αφού έχουμε γνωρίσει τις εντολές Serial.read() για την ανάγνωση δεδομένων από το Serial Monitor και τη δομή switch...case για τη λήψη αποφάσεων, θα τα συνδυάσουμε. Στόχος είναι να στέλνουμε έναν συγκεκριμένο χαρακτήρα (π.χ., έναν αριθμό από το '0' έως το '7') μέσω του Serial Monitor, και το Arduino, λαμβάνοντας αυτή την εντολή, να αλλάζει το χρώμα του RGB LED στο αντίστοιχο προκαθορισμένο χρώμα.

Ο Κώδικας:

Ο παρακάτω κώδικας χρησιμοποιεί τους ίδιους ορισμούς χρωμάτων (R1,G1,B1 κ.λπ.) από τη Δραστηριότητα 5. Η κύρια αλλαγή βρίσκεται στη συνάρτηση loop(), όπου τώρα ελέγχουμε για εισερχόμενους χαρακτήρες και χρησιμοποιούμε τη switch για να επιλέξουμε το χρώμα.

```
// Ακίδες PWM για το RGB LED (ίδιες με Δραστηριότητα 5)
const int redPin = 9;    // Ακίδα για το Κόκκινο στοιχείο
const int greenPin = 10; // Ακίδα για το Πράσινο στοιχείο
const int bluePin = 11;  // Ακίδα για το Μπλε στοιχείο

// Ορισμός μερικών βασικών χρωμάτων (Τιμές R, G, B από 0 έως 255)
// Χρώμα 1: Καθαρό Κόκκινο
const int valR1 = 255; const int valG1 = 0;   const int valB1 =
0;
// Χρώμα 2: Καθαρό Πράσινο
const int valR2 = 0;   const int valG2 = 255; const int valB2 =
0;
// Χρώμα 3: Καθαρό Μπλε
const int valR3 = 0;   const int valG3 = 0;   const int valB3 =
255;
// Χρώμα 4: Κίτρινο (Κόκκινο + Πράσινο)
const int valR4 = 255; const int valG4 = 255; const int valB4 =
0;
// Χρώμα 5: Μωβ/Ματζέντα (Κόκκινο + Μπλε)
const int valR5 = 255; const int valG5 = 0;   const int valB5 =
255;
// Χρώμα 6: Κυανό (Πράσινο + Μπλε)
const int valR6 = 0;   const int valG6 = 255; const int valB6 =
255;
// Χρώμα 7: Λευκό (Κόκκινο + Πράσινο + Μπλε)
const int valR7 = 255; const int valG7 = 255; const int valB7 =
255;
// Κατάσταση '0': Σβηστό LED
const int valR0 = 0;   const int valG0 = 0;   const int valB0 =
0;

void setup() {
  // Ορισμός των ακίδων ως εξόδους
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  Serial.begin(9600); // Έναρξη σειριακής επικοινωνίας
```

```
  Serial.println("===== Διαδραστικός Έλεγχος RGB LED =====");
  Serial.println("Εισάγετε έναν αριθμό (0-7) για να αλλάξετε
χρώμα:");
  Serial.println("1: Κόκκινο, 2: Πράσινο, 3: Μπλε, 4: Κίτρινο");
  Serial.println("5: Μωβ, 6: Κυανό, 7: Λευκό, 0: Σβηστό");
  Serial.println("----------------------------------------");

  // Ξεκινάμε με το LED σβηστό (Χρώμα 0)
  analogWrite(redPin, valR0);
  analogWrite(greenPin, valG0);
  analogWrite(bluePin, valB0);
}

void loop() {
  // Έλεγχος αν υπάρχουν διαθέσιμα δεδομένα από το Serial Monitor
  if (Serial.available() > 0) {
    // Ανάγνωση του εισερχόμενου χαρακτήρα
    char command = Serial.read();

    Serial.print("Ελήφθη εντολή: '");
    Serial.print(command);
    Serial.println("'");

    // Αποφάσεις βάσει της εντολής χρησιμοποιώντας switch...case
    switch (command) {
      case '1': // Κόκκινο
        analogWrite(redPin, valR1); analogWrite(greenPin, valG1);
analogWrite(bluePin, valB1);
        Serial.println("Αλλαγή σε: Κόκκινο");
        break;
      case '2': // Πράσινο
        analogWrite(redPin, valR2); analogWrite(greenPin, valG2);
analogWrite(bluePin, valB2);
        Serial.println("Αλλαγή σε: Πράσινο");
        break;
      case '3': // Μπλε
        analogWrite(redPin, valR3); analogWrite(greenPin, valG3);
analogWrite(bluePin, valB3);
        Serial.println("Αλλαγή σε: Μπλε");
        break;
      case '4': // Κίτρινο
        analogWrite(redPin, valR4); analogWrite(greenPin, valG4);
analogWrite(bluePin, valB4);
        Serial.println("Αλλαγή σε: Κίτρινο");
        break;
      case '5': // Μωβ
        analogWrite(redPin, valR5); analogWrite(greenPin, valG5);
analogWrite(bluePin, valB5);
        Serial.println("Αλλαγή σε: Μωβ");
        break;
      case '6': // Κυανό
        analogWrite(redPin, valR6); analogWrite(greenPin, valG6);
analogWrite(bluePin, valB6);
        Serial.println("Αλλαγή σε: Κυανό");
        break;
      case '7': // Λευκό
        analogWrite(redPin, valR7); analogWrite(greenPin, valG7);
analogWrite(bluePin, valB7);
        Serial.println("Αλλαγή σε: Λευκό");
```

```
        break;
      case '0': // Σβηστό
        analogWrite(redPin, valR0); analogWrite(greenPin, valG0);
analogWrite(bluePin, valB0);
        Serial.println("Αλλαγή σε: Σβηστό");
        break;
      case '\n':
        // Με '\n' συμβολίζεται το ENTER.
        break;
      default:
        // Αν ο χαρακτήρας δεν αντιστοιχεί σε καμία γνωστή εντολή
        Serial.println("Άγνωστη εντολή. Παρακαλώ εισάγετε έναν
αριθμό από 0 έως 7.");
        // Δεν αλλάζουμε το τρέχον χρώμα σε αυτή την περίπτωση.
        break;
    }
  }
}
```