Arduino Basics

Εφαρμογή 1: LED που αναβοσβήνει (Blink)

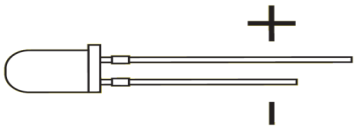
Στα πλαίσια της εφαρμογής θα παρουσιαστεί η χρήση των ψηφιακών ΠΙΝ της πλακέτας Arduino καθώς και η βασική δομή ενός προγράμματος Arduino. Η τελική κατασκευή θα είναι ένα LED που θα αναβοσβήνει περιοδικά.

Υλικά

Τα υλικά που θα χρησιμοποιήσουμε για την εφαρμογή μας:

LED

Το LED (Light Emitting Diode) είναι ένα στοιχείο, το οποίο όταν διαρρέεται από ρεύμα φωτοβολεί. Όσο μεγαλύτερη είναι η ένταση του ρεύματος που διαρρέει το LED, τόσο εντονότερο είναι το φως που παράγεται.



Το LED

Ως δίοδος, το LED επιτρέπει τη διέλευση του ρεύματος μόνο προς μία φορά. Όπως φαίνεται και στην Εικόνα 3 οι δύο ακροδέκτες του LED έχουν διαφορετικό μήκος. Ο πιο μακρύς ονομάζεται άνοδος και συνδέεται στο θετικό πόλο της πηγής (+), ενώ ο πιο κοντός ονομάζεται κάθοδος και συνδέεται στον αρνητικό πόλο (- ή GND ή γείωση).

Αντιστάτης

Ανάμεσα στο LED και την πηγή πρέπει να παρεμβάλλεται ένας αντιστάτης, προκειμένου να περιοριστεί το ρεύμα που θα διαρρεύσει το κύκλωμα και να προστατευτεί το LED. Η πιο συνηθισμένη τιμή αντίστασης που χρησιμοποιείται με το LED στις εφαρμογές Arduino είναι 220 Ω. (Μπορούμε όμως να βάλουμε οποιαδήποτε τιμή από 1 kΩ (1000 Ω) ως 180 Ω. Μεγάλη αντίσταση = μικρότερη φωτεινότητα του LED, αλλά και μικρότερη κατανάλωση ενέργειας, η οποία βέβαια είναι αμελητέα στις δικές μας εφαρμογές.

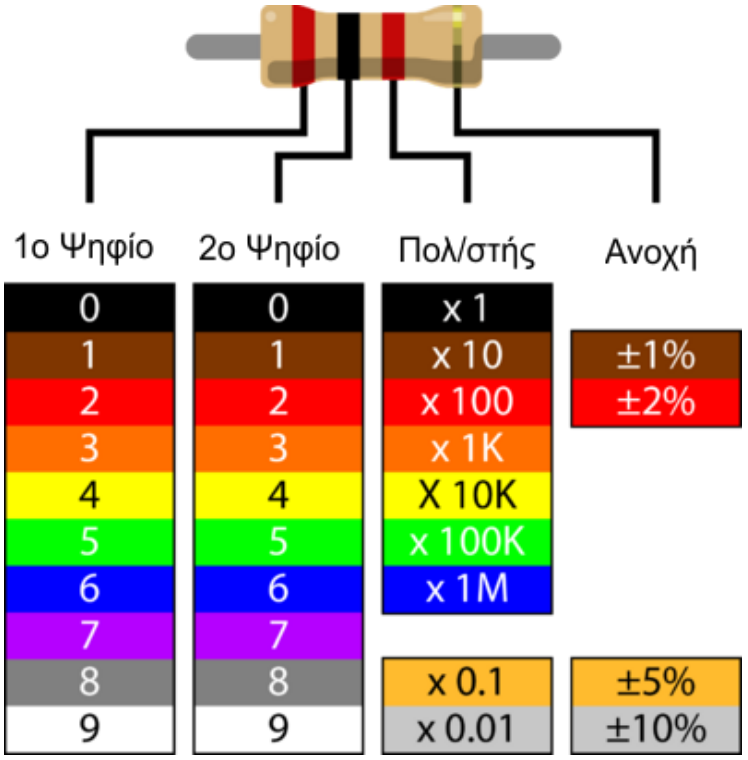
Αντιστάτης 220 Ω

Χρωματικός κώδικας αντιστατών

Η τιμή της αντίστασης ενός αντιστάτη δηλώνεται με χρωματιστές ζώνες επάνω στο εξάρτημα. Για τους αντιστάτες με μπεζ χρώμα, που είναι και οι πιο συνηθισμένοι, Η 1η και η 2η ζώνη είναι αριθμοί και η 3η ζώνη είναι ο πολλαπλασιαστής, δηλαδή μας λέει

πόσα μηδενικά να προσθέσουμε στο τέλος. Η 4η ζώνη, λιγότερο σημαντική για εμάς, δείχνει την ανοχή του αντιστάτη (ακρίβεια της τιμής του).

ΧΡΩΜΑΤΙΚΟΣ ΚΩΔΙΚΑΣ ΑΝΤΙΣΤΑΤΩΝ



Π.χ. για τον αντιστάτη του διπλανού σχήματος είναι:

1η ζώνη: Κόκκινο = 2

2η ζώνη: Κόκκινο = 2

3η ζώνη: Καφέ = 1 (Σημαίνει: Προσθέτω 1 μηδενικό)

Επομένως σχηματίζεται ο αριθμός: 2 2 0 δηλαδή 220 Ω.

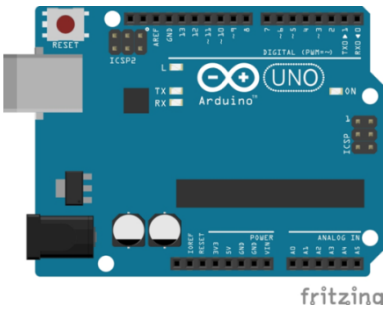
4η ζώνη: Χρυσό = Ανοχή 5%

Καλώδια

Για τη διασύνδεση των διαφόρων στοιχείων των κυκλωμάτων μας, θα χρησιμοποιηθούν ειδικά καλώδια, που είναι κατάλληλα για χρήση με breadboard και ονομάζονται jumper cables.

Arduino UNO

Το Arduino UNO διαθέτει 14 ψηφιακούς ακροδέκτες εισόδου/εξόδου (0 – 13). Όταν οι ψηφιακοί ακροδέκτες χρησιμοποιούνται ως έξοδοι, μπορούν να τεθούν σε μία από δύο καταστάσεις: HIGH (5V) και LOW (0V).

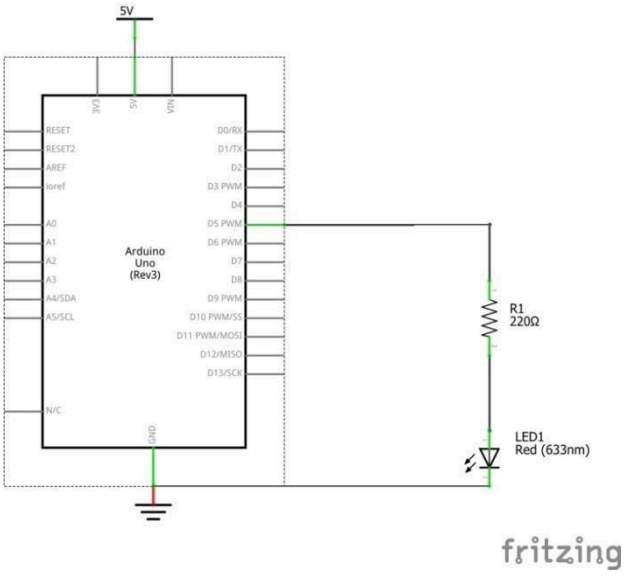


Arduino UNO

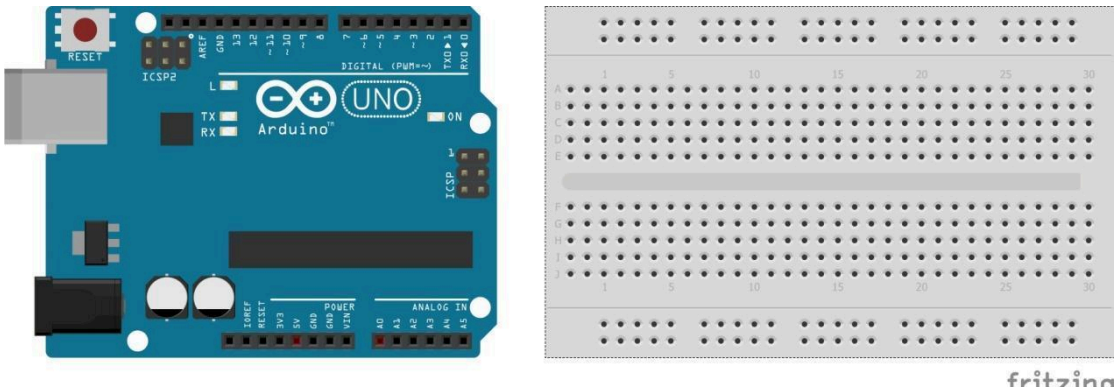
Στα πλαίσια της εφαρμογής μας θα χρησιμοποιήσουμε το Arduino ως μία προγραμματιζόμενη πηγή για την τροφοδοσία του κυκλώματος. Συγκεκριμένα θα αξιοποιήσουμε έναν από τους ψηφιακούς ακροδέκτες, ο οποίος όταν τίθεται σε κατάσταση HIGH το LED θα ανάβει, ενώ όταν τίθεται σε κατάσταση LOW το LED θα σβήνει. Ο ακροδέκτης γείωσης (GND) θα παίζει το ρόλο του αρνητικού πόλου της πηγής.

Κατασκευή κυκλώματος

Στη συνέχεια παρουσιάζεται βήμα προς βήμα η κατασκευή του κυκλώματος. Το ΣΧΗΜΑΤΙΚΟ ΔΙΑΓΡΑΜΜΑ του κυκλώματος που θα κατασκευάσουμε είναι:

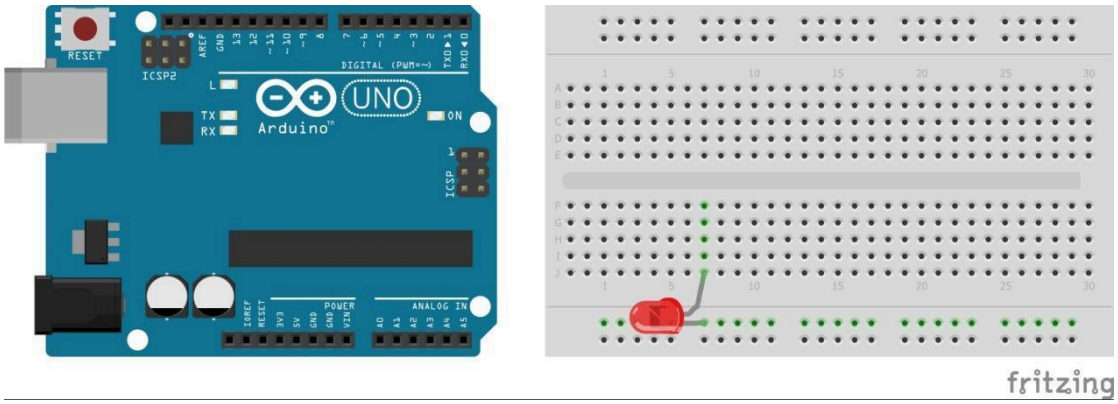


Ξεκινάμε με το Arduino UNO και ένα breadboard μισού μεγέθους όπως φαίνεται στην εικόνα. Ακολουθούμε τα βήματα που περιγράφονται στη συνέχεια.



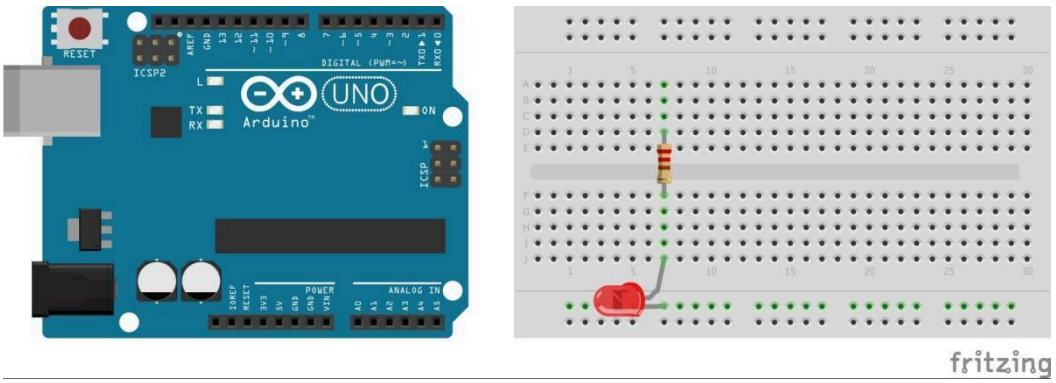
Βήμα 1

Συνδέουμε ένα LED στο breadboard όπως φαίνεται στο παρακάτω σχήμα. Προσέχουμε η ΑΝΟΔΟΣ (+) του LED (μακρύ πόδι) να είναι επάνω όπως βλέπουμε το σχήμα. Η ΚΑΘΟΔΟΣ (-) του LED (κοντό πόδι) συνδέεται κάτω, στη ράγα με τις πολλές οπές.



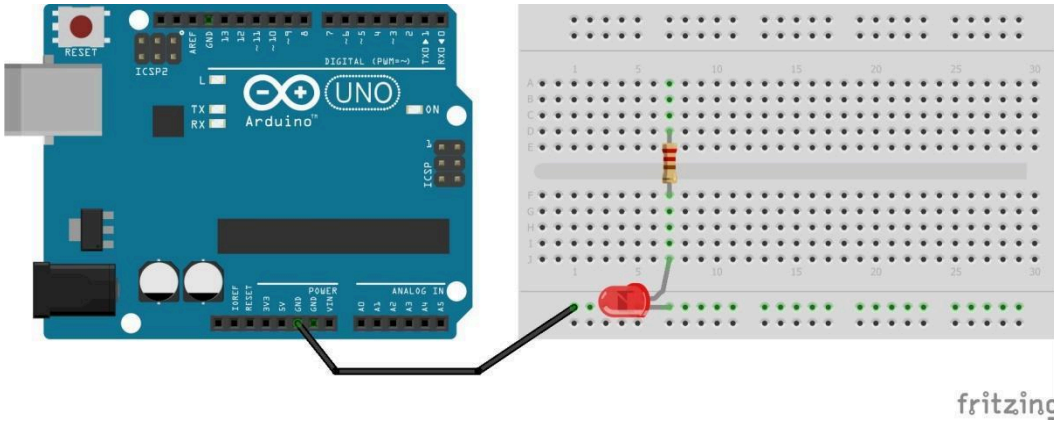
Βήμα 2

Συνδέουμε έναν αντιστάτη 220 Ω (κόκκινο, κόκκινο, καφέ, χρυσό) όπως φαίνεται στο σχήμα που ακολουθεί. Παρατηρούμε ότι συνδέεται η ΑΝΟΔΟΣ του LED με το ένα άκρο του αντιστάτη αφού μπήκαν στην ίδια πεντάδα οπών.



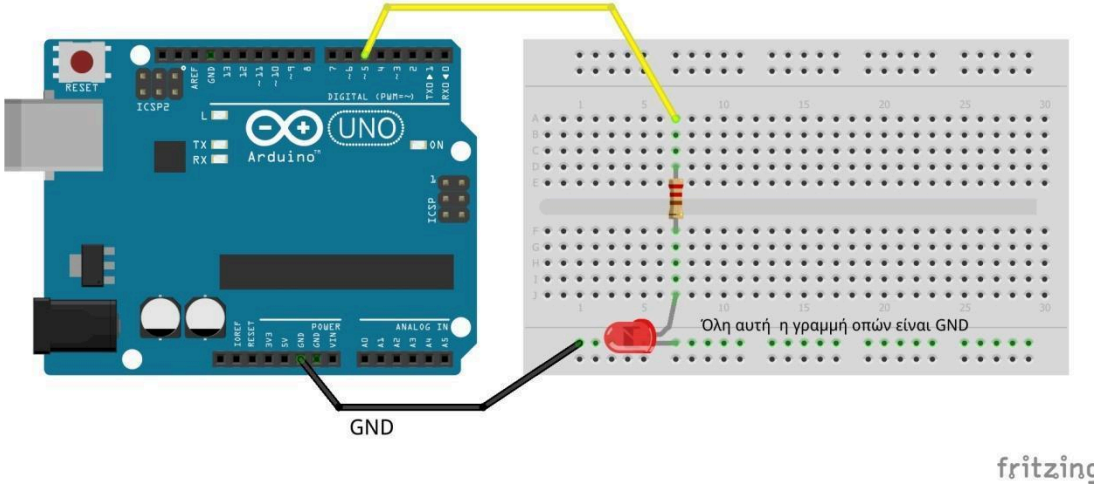
Βήμα 3

Συνδέουμε ένα καλώδιο από το πιν GND στην ακριανή οπή της γραμμής οπών όπου έχουμε συνδέσει την ΚΑΘΟΔΟ του LED. Έτσι, τώρα η ΚΑΘΟΔΟΣ του LED είναι συνδεμένη με τη γείωση (GND).



Βήμα 4

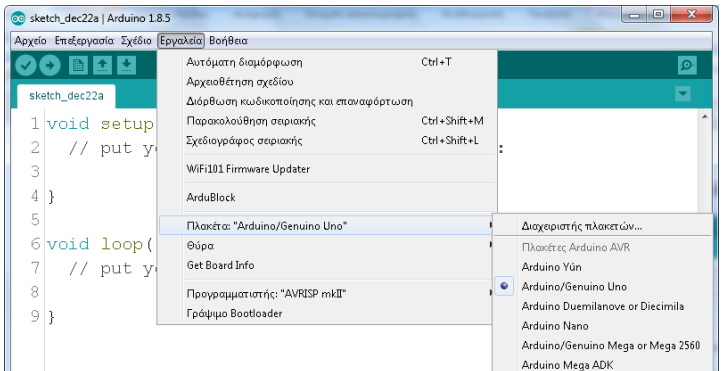
Συνδέουμε ένα καλώδιο από το ΨΗΦΙΑΚΟ ΠΙΝ 5 του Arduino στην πεντάδα οπών όπου είχαμε βάλει το ελεύθερο (επάνω) άκρο του αντιστάτη.



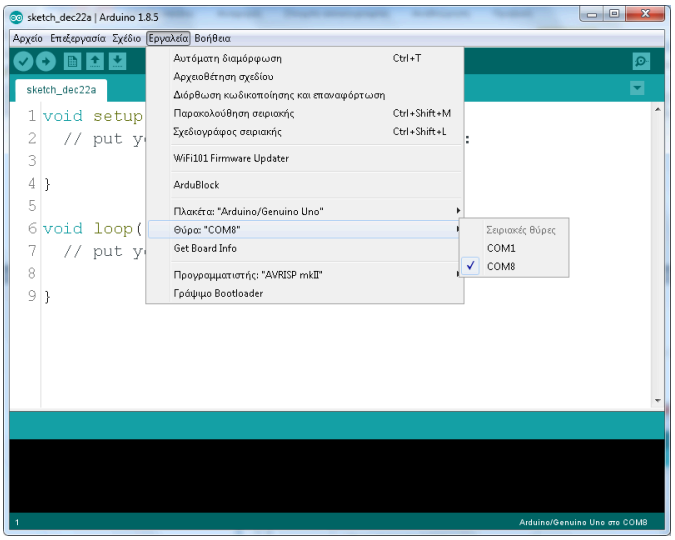
Τώρα το κύκλωμα είναι ολοκληρωμένο.

Σύνδεση του Arduino UNO με τον υπολογιστή

Συνδέουμε το Arduino UNO με το καλώδιο USB σε μία από τις USB θύρες του υπολογιστή. Στη συνέχεια, εκτελούμε το Arduino IDE. Στο παράθυρο που θα ανοίξει πηγαίνουμε στο μενού *Εργαλεία*, στην εγγραφή *Πλακέτα* και επιλέγουμε *Arduino/Genuino UNO*.



Ακολούθως, πάλι από το μενού *Εργαλεία*, πηγαίνουμε στο *Θύρα* και επιλέγουμε τη θύρα COM στην οποία έχει συνδεθεί το Arduino



Ανάπτυξη προγράμματος σε Arduino IDE

Τώρα βρισκόμαστε στο παράθυρο με το μεγάλο λευκό άδειο χώρο όπου γράφουμε το πρόγραμμα σε γλώσσα C (ακριβέστερα C++) του Arduino. Τα προγράμματα που γράφουμε ονομάζονται «σκίτσα» στην ορολογία του Arduino (sketches).

Κάθε πρόγραμμα έχει δύο διαδικασίες (υποπρογράμματα ή ομάδες εντολών) που προϋπάρχουν και ΕΙΝΑΙ ΑΠΑΡΑΙΤΗΤΕΣ στο πρόγραμμά μας: τις **setup()** και **loop().**

- Στη **setup** βάζουμε τις εντολές που θέλουμε να εκτελεστούν μία φορά μόνο.
- Στη **loop** βάζουμε τις εντολές που θέλουμε να επαναλαμβάνονται, αφού, όταν τελειώσει, η loop ξαναρχίζει από την αρχή της. Αυτό συνεχίζεται μέχρι να αποσυνδέσουμε το Arduino από την τάση τροφοδοσίας ή να πατήσουμε το κουμπί Reset.

Στο πρώτο παράδειγμα, γράφουμε ένα πρόγραμμα που αναβοσβήνει ένα LED. Το πρόγραμμά μας λέγεται «LED που αναβοσβήνει» ή "BLINK".

Το πρόγραμμα "BLINK" σε κώδικα C του Arduino:

```
void setup() {
  pinMode(5, OUTPUT);    // Όρισε το πιν 5 ως έξοδο
}

void loop() {
  digitalWrite(5, HIGH);   // Άναψε το LED
  delay(1000);                 // Περίμενε 1 δευτερόλεπτο εδώ
  digitalWrite(5, LOW);    // Σβήσε το LED
  delay(1000);                 // Περίμενε 1 δευτερόλεπτο εδώ
}
```

Οι εντολές που χρησιμοποιήθηκαν είναι:

- pinMode (… αριθμός του πιν , … INPUT ή OUTPUT);

Αυτή η εντολή ορίζει κάποιο πιν του Arduino ως ΕΙΣΟΔΟ (INPUT) ή ΕΞΟΔΟ (OUTPUT). ΣΗΜΕΙΩΣΗ: Τελειώνουμε την εντολή με το ελληνικό ερωτηματικό. Π.χ pinMode(5, OUTPUT);

- digitalWrite(… αριθμός του πιν, … HIGH ή LOW);

Αυτή η εντολή κάνει το αντίστοιχο πιν να βγάζει + (5V) ή - (GND). Ο αριθμός του πιν μπορεί να είναι από 0 ως 13 (αναφερόμαστε στα ψηφιακά πιν). Π.χ. digitalWrite(5, HIGH);

- delay(… χρόνος σε ms);

Αυτή η εντολή απλά σταματάει τη ροή του προγράμματος για ορισμένο χρόνο, τον οποίο δώσαμε σε χιλιοστά του δευτερολέπτου (millisecond ή ms). Π.χ. μπορούμε να γράψουμε: delay(2000); για καθυστέρηση 2 δευτερόλεπτα.

- // Σχόλια

Αν βάλουμε δύο κάθετες τη μια δίπλα στην άλλη, ό,τι ακολουθεί σε εκείνη τη σειρά αγνοείται από τον μεταγλωττιστή. Πρόκειται για ΣΧΟΛΙΟ. Είναι σημείωση για εμάς, για να μας υπενθυμίζει τι κάνει κάθε εντολή όταν διαβάζουμε μετά από καιρό το πρόγραμμα εμείς οι ίδιοι.

ΠΡΟΣΟΧΗ: Κάθε εντολή ΠΡΕΠΕΙ να τελειώνει με το ελληνικό ερωτηματικό. (Εξαιρούνται οι εντολές επανάληψης και επιλογής, που εδώ δεν χρησιμοποιήθηκαν).

Optional: Εφαρμογή 1: Βομβητής στη θέση του LED

Βομβητής (Active Buzzer)

Ο βομβητής ή buzzer είναι συσκευή που παράγει ήχο. Υπάρχουν δύο κατηγορίες buzzer, τα ενεργά (active) και τα παθητικά (passive). Τα active buzzer διαθέτουν εσωτερικό ταλαντωτή και όταν τροφοδοτούνται με συνεχή τάση, παρέχουν ένα τόνο συγκεκριμένης συχνότητας. Αντίθετα, τα passive buzzer δεν διαθέτουν εσωτερικό ταλαντωτή και για να παράγουν ήχο, πρέπει η τάση τροφοδοσίας τους να μεταβάλλεται (π.χ. HIGH□LOW□HIGH□LOW …). Η συχνότητα του ήχου που παράγει ένα passive buzzer είναι ίση με τη συχνότητα με την οποία μεταβάλλεται η τάση τροφοδοσίας του και άρα μπορεί να αλλάζει.

Βομβητής (Buzzer)

Πειραματισμός

Πραγματοποιούμε την δραστηριότητα με το λαμπάκι όπως περιεγράφηκε πριν, και μετά αντικαθιστούμε το λαμπάκι με το buzzer. Το πρόγραμμα που είναι περασμένο στην πλακέτα Arduino παραμένει το ίδιο. Απλά αλλάζουμε το LED με τον βομβητή και παρατηρούμε τον διακοπτόμενο ήχο που παράγεται.

Εφαρμογή 2: LED που ανάβει με το πάτημα κουμπιού

Οι ψηφιακοί ακροδέκτες του Arduino (ακροδέκτες 0 - 13 στο UNO) μπορούν να χρησιμοποιηθούν είτε ως ΕΞΟΔΟΙ είτε ως ΕΙΣΟΔΟΙ. Για να γνωρίσουμε τη χρήση των πιν ως ψηφιακών εισόδων του Arduino, θα προσθέσουμε στην 1η εφαρμογή ένα κουμπί συνδεμένο σε ένα άλλο πιν. Όσο το κουμπί παραμένει πατημένο, το LED θα είναι αναμμένο. Μόλις αφήνουμε το κουμπί, το LED θα σβήνει. Για το σκοπό αυτό, θα χρησιμοποιήσουμε μέσα στο πρόγραμμα μία εντολή ελέγχου: if(…) … else … (δηλαδή: *αν(…) …αλλιώς …)*, μέσα στην οποία θα ελέγχουμε την κατάσταση του πιν ΕΙΣΟΔΟΥ (πιν 2) και θα ενεργοποιούμε κατάλληλα το πιν ΕΞΟΔΟΥ (πιν 5).

Υλικά

Στη συνέχεια παρουσιάζονται τα επιπλέον υλικά που θα χρησιμοποιηθούν στη δεύτερη εφαρμογή.

Κουμπί

Στην εφαρμογή αυτή χρησιμοποιούμε για πρώτη φορά ένα κουμπί πίεσης (button).

(α) **(β)**

Όπως φαίνεται στην Εικόνα (α), το κουμπί πίεσης διαθέτει 4 ακροδέκτες. Οι ακροδέκτες αυτοί είναι ανά 2 συνδεδεμένοι μεταξύ τους (πάνω-κάτω), ενώ η διάταξη χωρίζεται σε δύο ανεξάρτητα κομμάτια (δεξί – αριστερό) όπως φαίνεται στο σχηματικό σύμβολο (Εικόνα (β)). Όταν πιέζουμε το κουμπί, κλείνει ο διακόπτης και συνδέεται το δεξί με το αριστερό του μέρος.

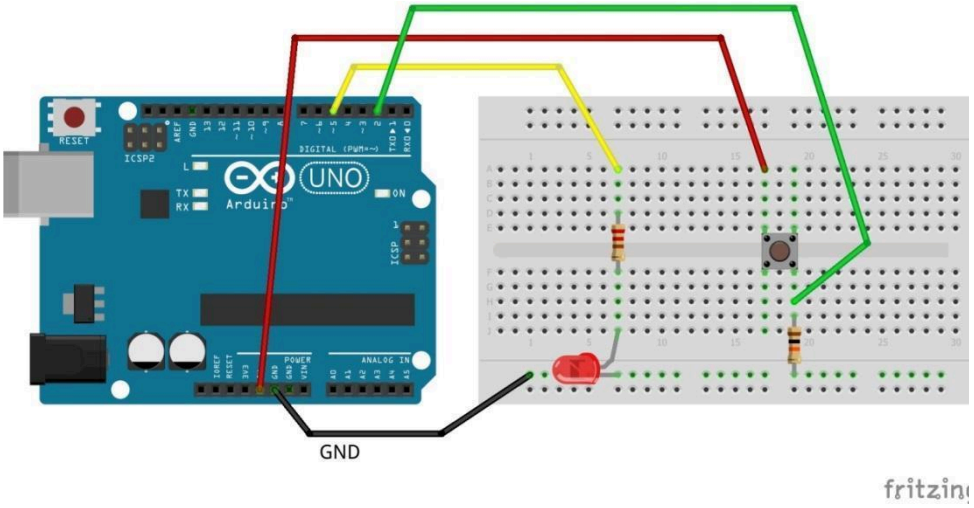
Αντιστάτης 10kΩ (χρώματα: καφέ, μαύρο, πορτοκαλί, χρυσό)

Για τη σύνδεση του κουμπιού με το Arduino θα χρησιμοποιήσουμε έναν αντιστάτη 10 kΩ.

Αντιστάτης 10kΩ

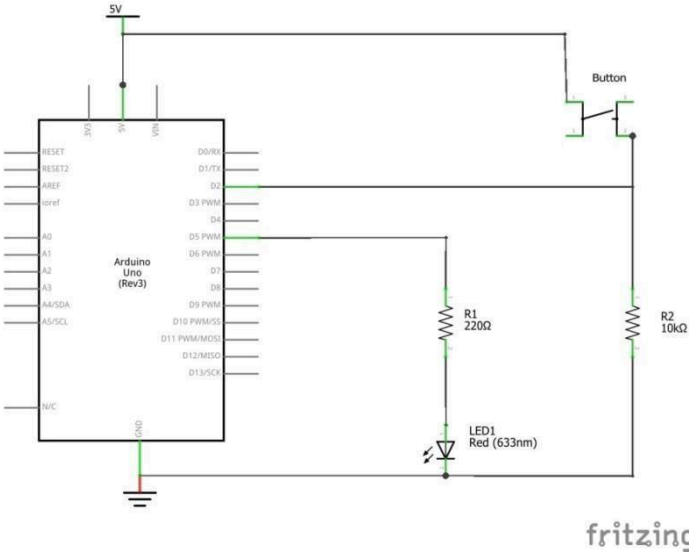
Κύκλωμα

Το κύκλωμα θα περιλαμβάνει ένα LED και ένα κουμπί. Όσον αφορά τη συνδεσμολογία του LED, δεν υπάρχει κάποια διαφοροποίηση σε σχέση με το κύκλωμα της 1ης εφαρμογής. Προσθέτουμε όμως το κουμπί (button) και έναν αντιστάτη 10kΩ. Κατασκευάζουμε το κύκλωμα όπως στο σχήμα:



Το αριστερό τμήμα του κουμπιού συνδέεται στην τροφοδοσία (pin 5 V) και το δεξί στη γείωση (GND) μέσω της αντίστασης 10 kΩ, για αποφυγή βραχυκυκλώματος όταν πατάμε το κουμπί. Για τον έλεγχο της κατάστασης του κουμπιού, συνδέουμε το δεξί του τμήμα στον ψηφιακό ακροδέκτη 2 του Arduino, που θα χρησιμοποιηθεί ως είσοδος. Όταν το κουμπί δεν είναι πατημένο, το κύκλωμα είναι ανοικτό, δεν υπάρχει ρεύμα, ούτε και πτώση τάσης στην αντίσταση των 10 kΩ. Άρα, ο ακροδέκτης 2 είναι συνδεμένος, μέσω της R2, στη γείωση και είναι σε δυναμικό 0V (κατάσταση LOW). Όταν πατηθεί το κουμπί, κλείνει το κύκλωμα και ο ακροδέκτης 2 βρίσκεται συνδεμένος σε δυναμικό 5 V, δηλαδή στον θετικό πόλο της τάσης τροφοδοσίας (κατάσταση HIGH).

Το σχηματικό διάγραμμα του κυκλώματος είναι:



Ανάπτυξη προγράμματος σε Arduino IDE

Το πρόγραμμα «LED και κουμπί» σε κώδικα C του Arduino:

```
void setup() {
  pinMode(5, OUTPUT);   // Όρισε το πιν 5 ως έξοδο  (LED)
  pinMode(2, INPUT);   // Όρισε το πιν 2 ως είσοδο  (Διακόπτης)
}

void loop() {
  if (digitalRead(2) == HIGH) {    // Αν ο διακόπτης είναι πατημένος…
    digitalWrite(5, HIGH);   // Άναψε το LED
  }
  else {                           // … αλλιώς …
    digitalWrite(5, LOW);   // Σβήσε το LED
  }
}
```

Η νέα εντολή είναι:

• digitalRead(… πιν);

Διαβάζει την κατάσταση του πιν που αναφέρουμε στην παρένθεση. Π.χ. η εντολή: a=digitalRead(2); διαβάζει την κατάσταση του πιν 2 και την αποθηκεύει στη μεταβλητή a.

ΠΑΡΑΤΗΡΗΣΗ: Μπορεί κάποιος να θέσει το ερώτημα: «Γιατί να μη συνδέσουμε απλά το κουμπί σε σειρά με τον αντιστάτη και το LED και να ξεφορτωθούμε και το Arduino; Πάλι θα λειτουργεί το κύκλωμα με τον ίδιο τρόπο.». Αυτό είναι σωστό, αλλά έχοντας ως «ενδιάμεσο» το Arduino, μπορούμε να κάνουμε αυτοματισμούς όπως; Να πατάμε το κουμπί στιγμιαία και το LED να ανάβει για ορισμένο χρόνο και μετά να σβήνει, όπως στο φως ενός κλιμακοστασίου πολυκατοικίας. Αυτό και πολλά άλλα μπορούν να γίνουν μόνο αν έχουμε στη διάθεσή μας ένα μικρό υπολογιστή όπως το Arduino. Χωρίς να αλλάξουμε το υλικό (hardware) μπορούμε να αλλάξουμε τον τρόπο λειτουργίας με αλλαγές στο λογισμικό (software).

Εφαρμογή 3: LED που αναβοσβήνει και LED με κουμπί

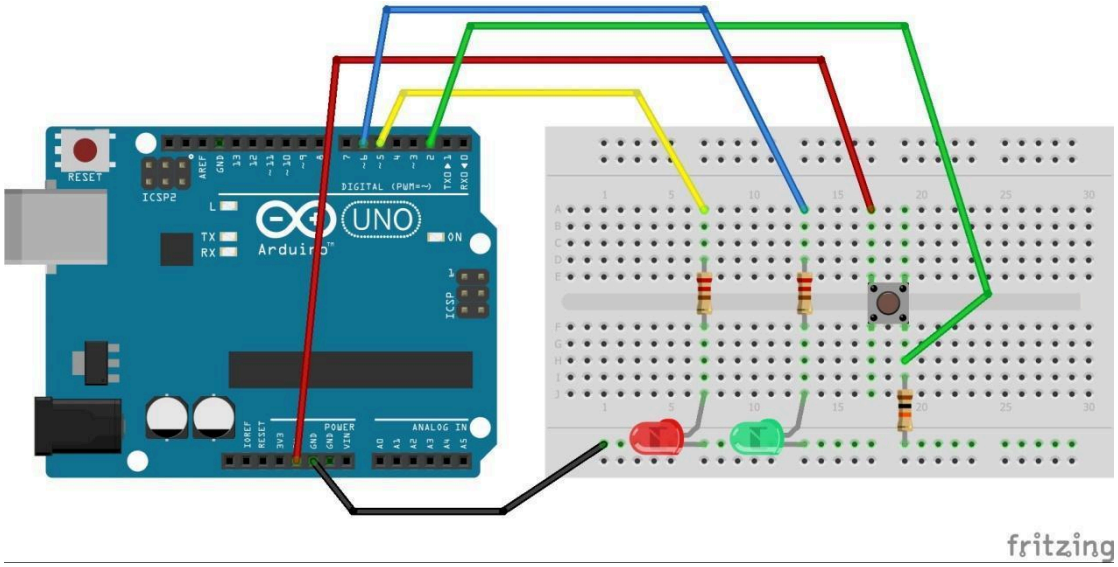
Αυτή η εφαρμογή αποτελεί έναν συνδυασμό των Εφαρμογών 1 και 2. Συγκεκριμένα, θα υπάρχουν δύο LED και ένα κουμπί. Το ένα LED θα αναβοσβήνει περιοδικά (όπως στην Εφαρμογή 1), ενώ το δεύτερο θα ανάβει όσο το κουμπί είναι πατημένο (Εφαρμογή 2). Στα πλαίσια της ανάπτυξης του προγράμματος, θα εντοπίσουμε ένα μειονέκτημα της εντολής *delay MILLIS* και θα προτείνουμε μία εναλλακτική υλοποίηση χρονοκαθυστέρησης.

Υλικά

Τα υλικά που θα χρησιμοποιηθούν για την εφαρμογή περιλαμβάνουν 2 LED (κόκκινο και πράσινο), 2 αντιστάσεις 220Ω (για τα LED), ένα κουμπί, 1 αντίσταση 10kΩ (για το κουμπί) και 5 καλώδια.

Κύκλωμα

Το τελικό κύκλωμα φαίνεται στην παρακάτω εικόνα



Όπως φαίνεται στην εικόνα, η άνοδος του δεύτερου LED έχει συνδεθεί, μέσω της αντίστασης 220Ω, στον ψηφιακό ακροδέκτη 6 του Arduino.

Σύνδεση πλακέτας Arduino UNO με τον υπολογιστή

Συνδέουμε την πλακέτα στον υπολογιστή και μέσα από το μενού *Εργαλεία* του Arduino IDE ορίζουμε τον τύπο της πλακέτας και τη θύρα σύνδεσης.

Ανάπτυξη προγράμματος σε Arduino IDE

Υλοποίηση_Α:

Το πρόγραμμα σε κώδικα C του Arduino:

```
void setup() {
  pinMode(5, OUTPUT);   // Όρισε το πιν 5 ως έξοδο  (κόκκινο LED)
  pinMode(6, OUTPUT);   // Όρισε το πιν 6 ως έξοδο  (πράσινο LED)
  pinMode(2, INPUT);    // Όρισε το πιν 2 ως είσοδο  (διακόπτης)
}


void loop() {
  if (digitalRead(2)== HIGH) {    // Αν ο διακόπτης είναι πατημένος…
    digitalWrite(5, HIGH);   // Άναψε το κόκκινο LED
  }
  else {                           // … αλλιώς …
    digitalWrite(5, LOW);    // Σβήσε το κόκκινο LED
  }
  digitalWrite(6, HIGH);   // Άναψε το πράσινο LED
  delay(3000);                  // Περίμενε 3 δευτερόλεπτα εδώ
  digitalWrite(6, LOW);    // Σβήσε το πράσινο LED
  delay(1000);                  // Περίμενε 3 δευτερόλεπτα εδώ
}
```

Υλοποίηση_Β:

Το πρόγραμμα σε κώδικα C του Arduino:

```
unsigned long start_time; // Δημιούργησε μια μεταβλητή τύπου unsigned long integer με όνομα start_time


 void setup() {
  pinMode(5, OUTPUT);   // Όρισε το πιν 5 ως έξοδο  (κόκκινο LED)
  pinMode(6, OUTPUT);   // Όρισε το πιν 6 ως έξοδο  (πράσινο LED)
  pinMode(2, INPUT);    // Όρισε το πιν 2 ως είσοδο  (διακόπτης)
}


void loop() {
  digitalWrite(6, HIGH);   // Άναψε το πράσινο LED
  start_time=millis();     // Βάλε στην start_time την τιμή της millis()
```

```
  while (millis() - start_time < 3000) {  // Εφόσον δεν πέρασαν 3 δευτ κάνε:
    if (digitalRead(2) == HIGH) {    // Αν ο διακόπτης είναι πατημένος…
      digitalWrite(5, HIGH);   // Άναψε το κόκκινο LED
    }
    else {                           // … αλλιώς …
      digitalWrite(5, LOW);   // Σβήσε το κόκκινο LED
    }
  }   // (τέλος της while)
  digitalWrite(6, LOW);   // Σβήσε το πράσινο LED
  start_time=millis();    // Βάλε στην start_time την τιμή της millis()
  while (millis() - start_time < 3000) {  // Εφόσον δεν πέρασαν 3 δευτ κάνε:
    if (digitalRead(2) == HIGH) {    // Αν ο διακόπτης είναι πατημένος…
      digitalWrite(5, HIGH);   // Άναψε το κόκκινο LED
    }
    else {                           // … αλλιώς …
      digitalWrite(5, LOW);   // Σβήσε το κόκκινο LED
    }
  }   // (τέλος της while)
}  // (τέλος της loop)
```

Οι εντολές που χρησιμοποιούνται είναι:

• unsigned long start_time

Εδώ δηλώνουμε μια ακέραια μεταβλητή τύπου unsigned long integer (32 bit). Μπορεί να αποθηκεύσει αριθμό από 0 ως 4.294.967.295 (ο απλός long integer, πάλι 32 bit, μπορεί να πάρει τιμές από -2.147.483.648 έως 2.147.483.647). Επειδή η τιμή της millis()πάει από 0 ως 4.294.967.295, χρειάζεται να έχουμε μια μεταβλητή αντίστοιχου τύπου για να χωράει να αποθηκεύσουμε την τιμή της.

ΣΗΜ.: Είναι απαραίτητο να δηλωθεί μια μεταβλητή πριν χρησιμοποιηθεί. Αλλιώς, όταν πάμε να μεταγλωττίσουμε το πρόγραμμα, θα σταματήσει η μεταγλώττιση και θα εμφανιστεί μήνυμα σφάλματος.

• millis()

Είναι ένας αριθμός που αυξάνεται συνέχεια σε ένα εσωτερικό «χρονόμετρο» του Arduino. Ξεκινάει να μετράει από το 0 όταν δώσουμε ρεύμα στο Arduino και αυξάνει την τιμή του κατά 1 κάθε χιλιοστό του δευτερολέπτου (κάθε 1 ms). Ξαναμηδενίζεται όταν φτάσει την τιμή 4.294.967.295 (σε περίπου 50 ημέρες) ή αν πατήσουμε το κουμπί reset.

• while(… συνθήκη){ … εντολές }

Αυτή είναι μια δομή επανάληψης με συνθήκη, γνωστή σε όσους έχουν κάνει προγραμματισμό. Η συνθήκη εξετάζεται και εφόσον είναι αληθής εκτελούνται οι εντολές που υπάρχουν μέσα στα άγκιστρα. Στη συνέχεια η εκτέλεση ξαναπάει στην αρχή της while και ελέγχεται πάλι η συνθήκη. Αν δεν ισχύει η συνθήκη κάποια στιγμή, οι εντολές που υπάρχουν στις αγκύλες δεν εκτελούνται και το πρόγραμμα προχωράει παρακάτω, μετά το άγκιστρο τέλους.

• Η δομή ελέγχου if(… συνθήκη) {… εντολές} else {… άλλες εντολές} είναι γνωστή από πριν.

Σύγκριση των δύο υλοποιήσεων (Α και Β)

Όπως μπορείτε να παρατηρήσετε, ενώ το LED που αναβοσβήνει λειτουργεί κανονικά, το LED που ελέγχεται από το κουμπί δεν έχει την αναμενόμενη συμπεριφορά.

Η δυσλειτουργία που παρατηρείται **στην Α υλοποίηση**, οφείλεται στο γεγονός ότι η κατάσταση του κουμπιού ελέγχεται στο πρόγραμμα μόνο μία στιγμή στην αρχή (εντολή if… else…), ενώ στα 6 δευτερόλεπτα που διαρκεί το αναβόσβημα του δεύτερου LED, το κουμπί δεν ελέγχεται και άρα οποιαδήποτε αλλαγή στην κατάστασή του αγνοείται. Για να μπορέσει λοιπόν το κουμπί να λειτουργήσει σωστά, θα έπρεπε η κατάστασή του να ελέγχεται συνέχεια κατά τη διάρκεια των δύο χρονοκαθυστερήσεων (εντολές delay). Αυτό δε μπορεί να γίνει αν χρησιμοποιήσουμε τις "delay" γιατί πολύ απλά το πρόγραμμα ΣΤΑΜΑΤΑΕΙ όταν εκτελείται μια εντολή delay.

Η λύση είναι η χρήση των millis(): **Στη Β υλοποίηση** παρακολουθούμε σε κάθε επανάληψη της loop αν πέρασε ο επιθυμητός χρόνος πριν κάνουμε την επόμενη ενέργεια. Αυτό έχει το πλεονέκτημα ότι το πρόγραμμα τρέχει συνεχώς (ή μάλλον «γυρίζει συνεχώς») μέσα στη loop και μπορεί να εξετάζει σε κάθε επανάληψη την κατάσταση του κουμπιού και να ανάψει αν χρειάζεται το LED. Επομένως το πρόγραμμα μπορεί να αντιδράσει σε πολύ σύντομο χρόνο αν πατήσουμε το κουμπί.

Εφαρμογή 4 : Φανάρι κυκλοφορίας

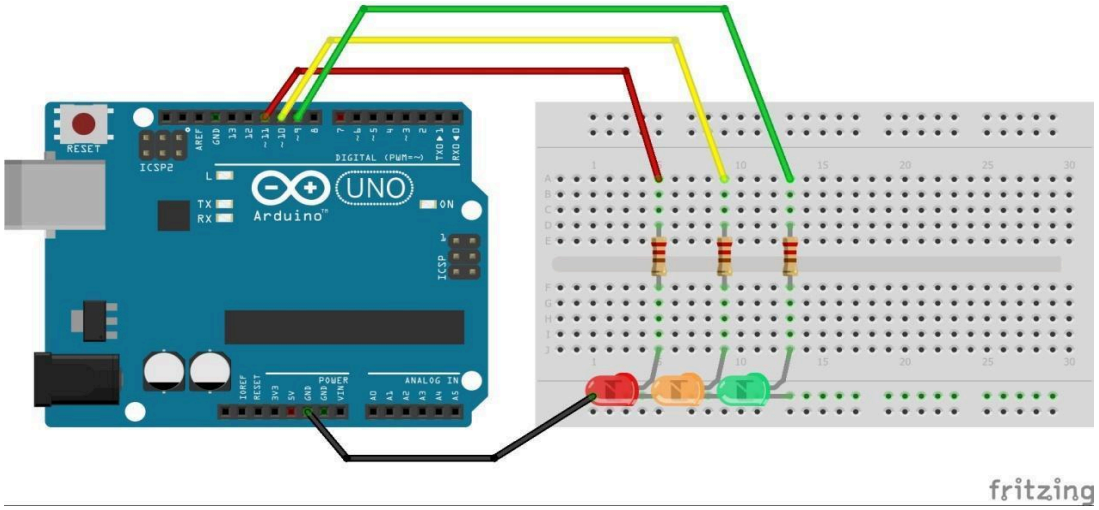
Η εφαρμογή που θα αναπτυχθεί θα προσομοιώνει τη λειτουργία ενός φαναριού κυκλοφορίας για αυτοκίνητα (μη ξεχνάτε ότι υπάρχουν και τα φανάρια πεζών). Συγκεκριμένα, θα χρησιμοποιηθούν τρία LED (κόκκινο, πορτοκαλί, πράσινο), τα οποία θα ανάβουν εναλλάξ με προκαθορισμένες διάρκειες. Στα πλαίσια της εφαρμογής, παρουσιάζεται η αξιοποίηση του breadboard σε πιο σύνθετα κυκλώματα και η έννοια του συγχρονισμού ενεργειών στο πρόγραμμα.

Υλικά

Θα χρησιμοποιηθούν: 1 κόκκινο LED, 1 πορτοκαλί ή κίτρινο LED, 1 πράσινο LED, 3 αντιστάσεις των 220 Ω και 4 καλώδια.

Κύκλωμα

Κατασκευάζουμε το κύκλωμα στο breadboard όπως δείχνει το παρακάτω σχήμα:



Στη συνέχεια, γράφουμε το παρακάτω πρόγραμμα στο Arduino IDE για να λειτουργήσουμε τα φανάρια:

Ανάπτυξη προγράμματος σε Arduino IDE

Το πρόγραμμα σε κώδικα C του Arduino:

```
int ledRed = 11;
int ledOrange = 10;
int ledGreen = 9;

void setup() {
pinMode(ledRed, OUTPUT);
pinMode(ledOrange, OUTPUT);
pinMode(ledGreen, OUTPUT);
}

void loop() {
// κόκκινο για 3 δευτερόλεπτα
digitalWrite(ledRed, HIGH);
digitalWrite(ledOrange, LOW);
digitalWrite(ledGreen, LOW);
delay(3000);

// πράσινο για 5 δευτερόλεπτα
digitalWrite(ledRed, LOW);
digitalWrite(ledOrange, LOW);
digitalWrite(ledGreen, HIGH);
delay(5000);
```

```
// πορτοκαλί για 1 δευτερόλεπτο
digitalWrite(ledRed, LOW);
digitalWrite(ledOrange, HIGH);
digitalWrite(ledGreen, LOW);
delay(1000);
}
```

Εντολές που χρησιμοποιούνται:

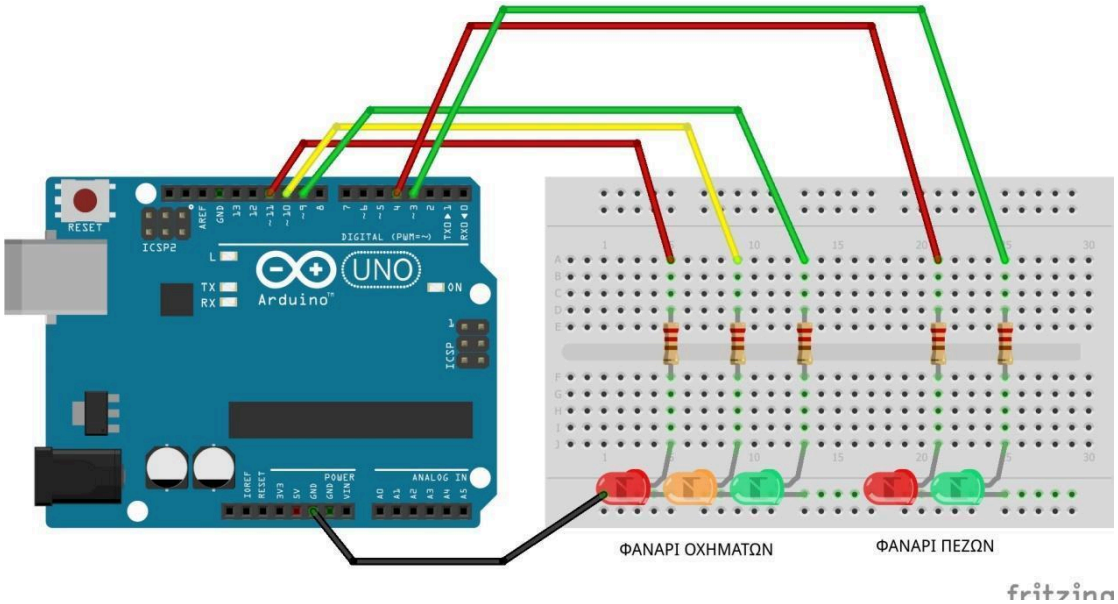
- Int …; Ορίζει μια ακέραιη μεταβλητή.

Optional: Εφαρμογή 2: Φανάρι κυκλοφορίας με φανάρι πεζών

Στην προηγούμενη εφαρμογή προσθέτουμε και ένα φανάρι για πεζούς.

Κύκλωμα

Κατασκευάζουμε το παρακάτω κύκλωμα βάζοντας επιπλέον 2 LED (κόκκινο και πράσινο) με αντιστάτες των 220Ω όπως δείχνει η παρακάτω εικόνα:



Στη συνέχεια, γράφουμε το παρακάτω πρόγραμμα στο Arduino IDE για να λειτουργήσουμε τα φανάρια πεζών:

Ανάπτυξη προγράμματος σε Arduino IDE

Το πρόγραμμα σε κώδικα C του Arduino:

```
int ledRed = 11;
int ledOrange = 10;
 int ledGreen = 9;
int pedRed = 4;
int pedGreen = 3;

void setup() {
pinMode(ledRed, OUTPUT);
pinMode(ledOrange, OUTPUT);
pinMode(ledGreen, OUTPUT);
 pinMode(pedRed, OUTPUT);
pinMode(pedGreen, OUTPUT);
}

void loop() {
// κόκκινο για 3 δευτερόλεπτα, πράσινο στους πεζούς
digitalWrite(ledRed, HIGH);
digitalWrite(ledOrange, LOW);
digitalWrite(ledGreen, LOW);
digitalWrite(pedRed, LOW);
 digitalWrite(pedGreen, HIGH);
 delay(3000);

// κόκκινο στους πεζούς, περιμένω για 1 δευτερόλεπτο πριν δώσω πράσινο στα αμάξια digitalWrite(pedRed, HIGH);
digitalWrite(pedGreen, LOW);
delay(1000);

// πράσινο κυκλοφορίας για 5 δευτερόλεπτα
digitalWrite(ledRed, LOW);
digitalWrite(ledOrange, LOW);
digitalWrite(ledGreen, HIGH);
delay(5000);

// πορτοκαλί για 1 δευτερόλεπτο
digitalWrite(ledRed, LOW);
digitalWrite(ledOrange, HIGH);
```

```
digitalWrite(ledGreen, LOW);
delay(1000);
}
```