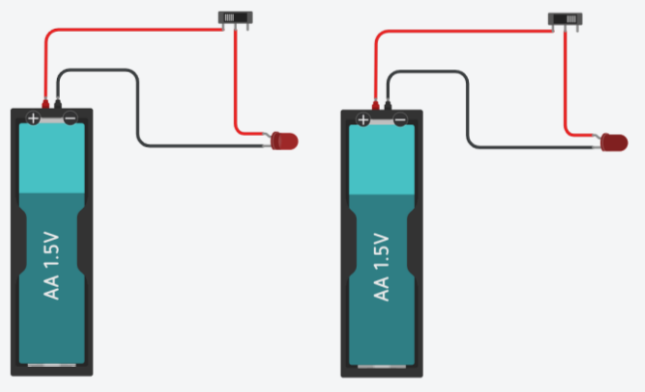
Εισαγωγή στον Προγραμματισμό με Arduino

Περιεχόμενα

Πίνακας εικόνων

Εισαγωγή

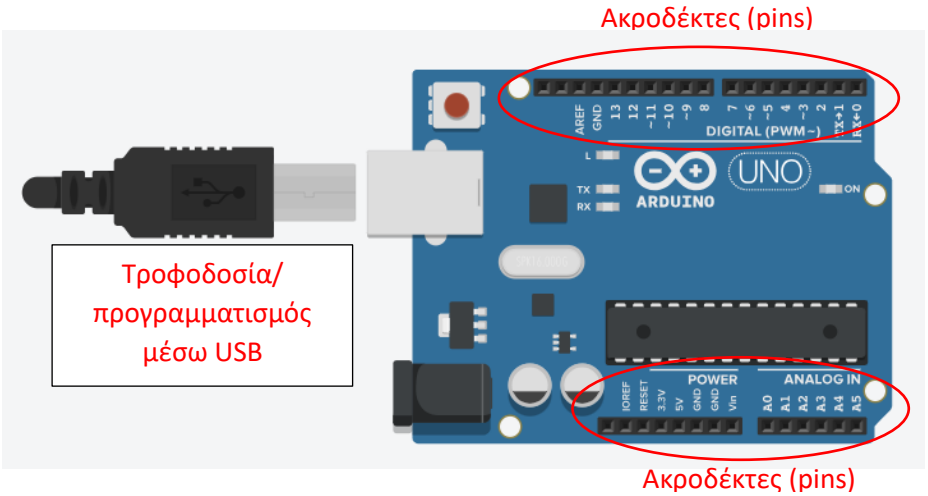
Από τα πιο απλά και βασικά μαθήματα στην εισαγωγή προγραμματισμού μικροελεγκτών είναι η οδήγηση (να μπορέσουμε αναβοσβήσουμε δηλαδή) μίας Διόδου Εκπομπής Φωτός, (LED – Light-Emitting Diode). Μέσα στην τάξη είδαμε πως μπορούμε να φτιάξουμε ένα βασικό ηλεκτρονικό κύκλωμα που να αναβοσβήνει μία δίοδο LED μέσω ενός διακόπτη. Μία αντίστοιχη διάταξη αναπαρίσταται στο κύκλωμα της **Εικόνας 1**. Μετακινώντας το διακόπτη στην αριστερή θέση ανάβει η δίοδος LED, ενώ στη δεξιά θέση σβήνει η δίοδος LED (αριστερή και δεξιά εικόνα αντιστοίχως).



Εικόνα 1 Ηλεκτρονικό κύκλωμα ελέγχου LED με διακόπτη

Η πλακέτα Arduino UNO

Παρακάτω παρουσιάζεται η πλακέτα Arduino UNO (**Εικόνα 2**). Στην πάνω και κάτω πλευρά αναγράφονται πληροφορίες σχετικές με τους ακροδέκτες της πλακέτας. Η πλακέτα Arduino UNO αποτελεί ένα πλήρες υπολογιστικό σύστημα, που όμως σε αντίθεση με τον προσωπικό υπολογιστή που γνωρίζουμε, προορίζεται για την υλοποίηση μιας ειδικής λειτουργίας (π.χ. σύστημα αυτόματου ποτίσματος μιας γλάστρας).



Εικόνα 2 Η πλακέτα Arduino UNO

Ενώ στον προσωπικό υπολογιστή έχουμε τις θύρες USB, HDMI, κλπ., στις οποίες συνδέουμε τις περιφερειακές μονάδες εισόδου/εξόδου (π.χ. πληκτρολόγιο, οθόνη, κλπ.), στην πλακέτα Arduino UNO, τις θύρες εισόδου/εξόδου αναλαμβάνουν οι ακροδέκτες. Για παράδειγμα

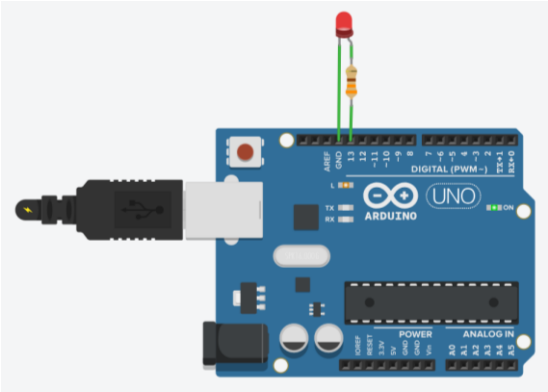
μπορούμε να συνδέσουμε μία απλή περιφερειακή μονάδα εξόδου, όπως είναι η δίοδος LED, ή μία απλή μονάδα εισόδου, όπως ένας διακόπτης.

Πιο σύνθετες περιφερειακές μονάδες εισόδου/εξόδου μπορεί να δεσμεύουν περισσότερους από έναν ακροδέκτες στην πλακέτα Arduino UNO, όπως π.χ. ένας αισθητήρας απόστασης σαν αυτούς που χρησιμοποιούνται στα αυτοκίνητα κατά το παρκάρισμα, για την ανίχνευση εμποδίου στο πίσω μέρος του αυτοκινήτου.

1η εφαρμογή με Arduino UNO: σύνδεση και οδήγηση LED

Στη συνέχεια θα παρουσιάσουμε τον πρώτο μας κώδικα που αναβοσβήνει αυτόματα μία δίοδο LED ανά ένα δευτερόλεπτο (second–sec). Το πρώτο βήμα, λοιπόν, είναι η διασύνδεση μιας LED με την πλακέτα του Arduino, όπως φαίνεται στο παρακάτω κύκλωμα. Στο συγκεκριμένο κύκλωμα η δίοδος LED οδηγείται από τον pin13 του μικροελεγκτή.

Το εξάρτημα που παρεμβάλλεται μεταξύ pin13 και LED, ονομάζεται αντίσταση και, όπως γνωστοποιεί το όνομά του, προβάλλει μία αντίσταση (περιορίζει δηλαδή) το ρεύμα που διέρχεται από τη δίοδο LED. Χωρίς την αντίσταση είναι πιθανό να καταστραφεί η δίοδος LED (ή ακόμη να προκληθεί μόνιμη βλάβη στο pin13 της πλακέτας).

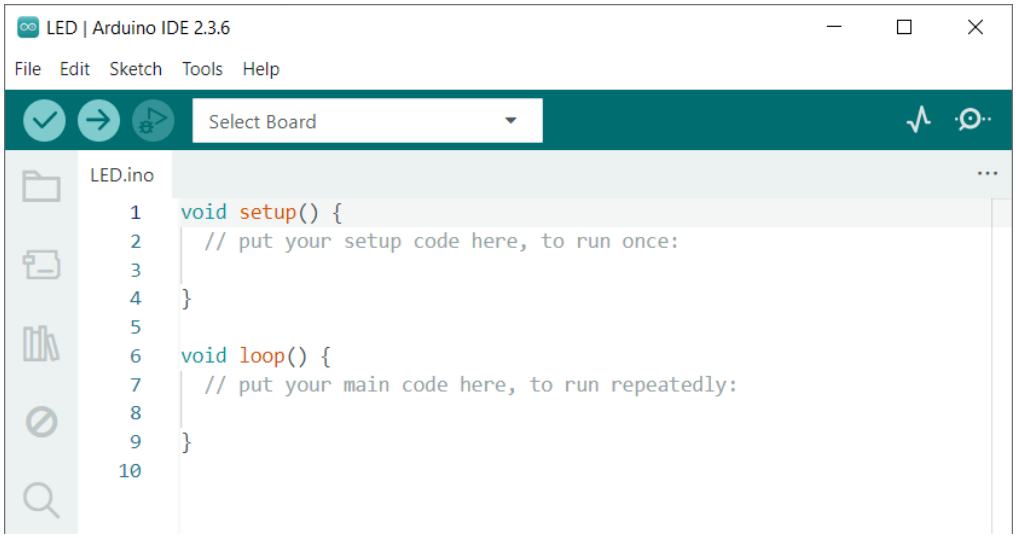


Εικόνα 3 Σύνδεση LED στην πλακέτα Arduino UNO

Κάθε πρόγραμμα που γράφουμε στο περιβάλλον Arduino IDE, είτε αναφέρεται στην πλακέτα Arduino UNO είτε σε άλλη πλακέτα, περιλαμβάνει υποχρεωτικά δύο συναρτήσεις, τη **setup()** και τη **loop()**. Όταν λοιπόν ξεκινάμε ένα νέο κώδικα στο περιβάλλον Arduino IDE δημιουργείται ένα «κενό» πρόγραμμα με τις δύο αυτές συναρτήσεις.

Μία συνάρτηση μπορεί να λαμβάνει τα λεγόμενα ορίσματα, τα οποία τοποθετούνται εντός των παρενθέσεων που συντάσσονται αμέσως μετά από το όνομα της συνάρτησης, ενώ όταν η συνάρτηση εκτελείται μπορεί να επιστρέφει μία τιμή. Για παράδειγμα, αν δημιουργούσαμε μία συνάρτηση για την πρόσθεση δύο αριθμών X και Y, οι αριθμοί αυτοί θα τοποθετούνταν εντός των παρενθέσεων και όταν εκτελούνταν η συνάρτηση, θα επέστρεφε το άθροισμα X+Y.

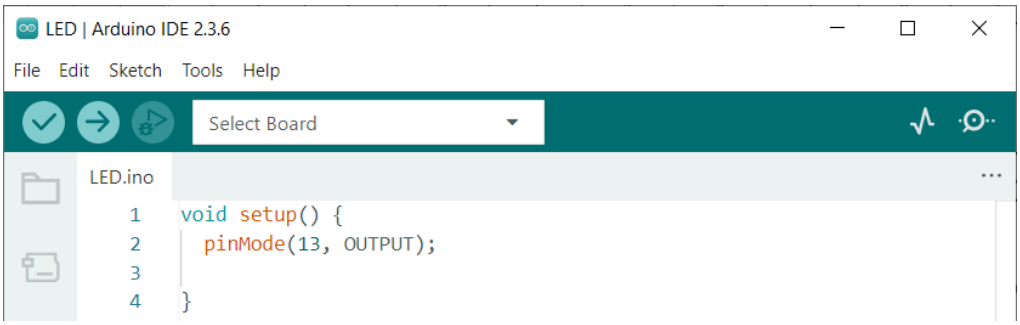
Οι συναρτήσεις **setup()** και **loop()** δε λαμβάνουν ορίσματα, συνεπώς δεν συντάσσεται κάτι μέσα στις παρενθέσεις των συναρτήσεων, και επίσης, δεν επιστρέφουν κάποια τιμή όταν εκτελούνται. Συνεπώς, η κωδική λέξη **void** που συντάσσεται ακριβώς πριν το όνομα της συνάρτησης (και στα ελληνικά μεταφράζεται ως **κενό**), προσδιορίζει αυτή ακριβώς την ιδιότητα της συνάρτησης, ότι δηλαδή η συνάρτηση δεν επιστρέφει κάποια τιμή όταν εκτελείται.



Εικόνα 4 Έναρξη νέου κώδικα στο περιβάλλον Arduino IDE (μενού: File → New Sketch)

Η συνάρτηση setup()

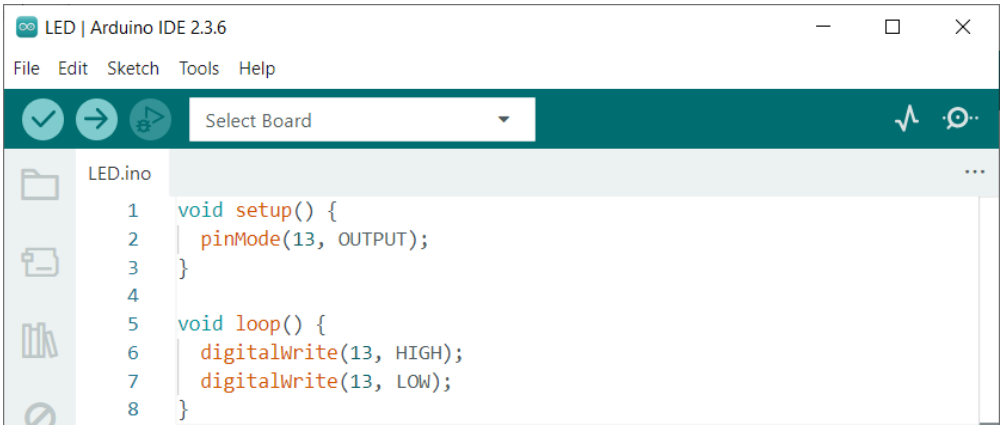
Η συνάρτηση **setup()** εκτελείται κατά την εκκίνηση λειτουργίας της πλακέτας Arduino UNO και είναι υπεύθυνη για τον ορισμό εισόδων και εξόδων. Όπως αναφέρθηκε προηγουμένως, οι ακροδέκτες της πλακέτες χρησιμοποιούνται για την διασύνδεση των περιφερειακών μονάδων εισόδου και εξόδου. Κάθε ακροδέκτης μπορεί να οριστεί είτε ως έξοδος, όταν για παράδειγμα χρησιμοποιείται για την οδήγηση ενός LED, είτε ως είσοδος (όταν π.χ. διαβάζει την τιμή ενός διακόπτη). Θα πρέπει όμως να οριστεί στο πρόγραμμα η ιδιότητα αυτή του ακροδέκτη.



Εικόνα 5 Ορισμός του pin13 ως ακροδέκτη εξόδου

Στην **Εικόνα 5** παρουσιάζεται η εντολή που ορίζει το pin13 ως ακροδέκτη εξόδου (καθότι σε αυτόν τον ακροδέκτη έγινε η σύνδεση της διόδου LED, δηλαδή μιας περιφερειακής μονάδας εξόδου). Η εντολή **pinMode()** αποτελεί μία συνάρτηση του Arduino IDE, η οποία λαμβάνει δύο ορίσματα που χωρίζονται με κόμμα. Το πρώτο όρισμα περιγράφει τον αριθμό του ακροδέκτη στον οποίο είναι συνδεδεμένη η δίοδος LED, ενώ το δεύτερο όρισμα περιγράφει την ιδιότητα που λαμβάνει ο ακροδέκτης. Συγκεκριμένα, η λέξη **OUTPUT** ορίζει το pin έξοδο, ενώ αν θέλαμε να ορίσουμε το pin είσοδο θα χρησιμοποιούσαμε τη λέξη **INPUT**[1].

[1] Θα πρέπει να τονιστεί ότι η σύνταξη εντολών πρέπει να χρησιμοποιείται όπως ακριβώς δίδεται στα παραδείγματα, καθότι υπάρχει διάκριση πεζών-κεφαλαίων γραμμάτων (π.χ. η λέξη **output** με μικρά γράμματα δεν αναγνωρίζεται ως ορθή εντολή από το Arduino IDE).



Εικόνα 6 Αναβόσβησμα της διόδου LED

Η συνάρτηση loop()

Στο σημείο αυτό έχουμε ολοκληρώσει τον ορισμό εισόδων/εξόδων της εφαρμογής που επιθυμούμε να υλοποιήσουμε και ξεκινάμε τον κυρίως κώδικα που αναβοσβήνει τη δίοδο LED, ο οποίος συντάσσεται εντός της συνάρτησης **loop()**. Ουσιαστικά, η loop() αποτελεί μία δομή ελέγχου της ακολουθιακής ροής του κώδικα, η λειτουργία της οποίας θα γίνει κατανοητή στη συνέχεια του κεφαλαίου.

Η εντολή οδήγησης της LED είναι η **digitalWrite()**, η οποία επίσης είναι συνάρτηση (επίσης δύο ορισμάτων) του Arduino. Όπως γνωστοποιεί το όνομά της, η εντολή αυτή γράφει (write) μία ψηφιακή (digital) τιμή σε έναν ακροδέκτη εξόδου. Στην εντολή της γραμμής 6, η **digitalWrite** γράφει στο pin13 τιμή **HIGH**, ενώ στην εντολή της γραμμής 7, η **digitalWrite** γράφει στο pin13 τιμή **LOW**. Με απλά λόγια θα μπορούσαμε να πούμε ότι, η πρώτη εντολή επιτρέπει τη διέλευση ρεύματος προς τη δίοδο LED οπότε και αυτή ανάβει, ενώ η δεύτερη εντολή δεν επιτρέπει τη διέλευση ρεύματος οπότε η LED σβήνει.

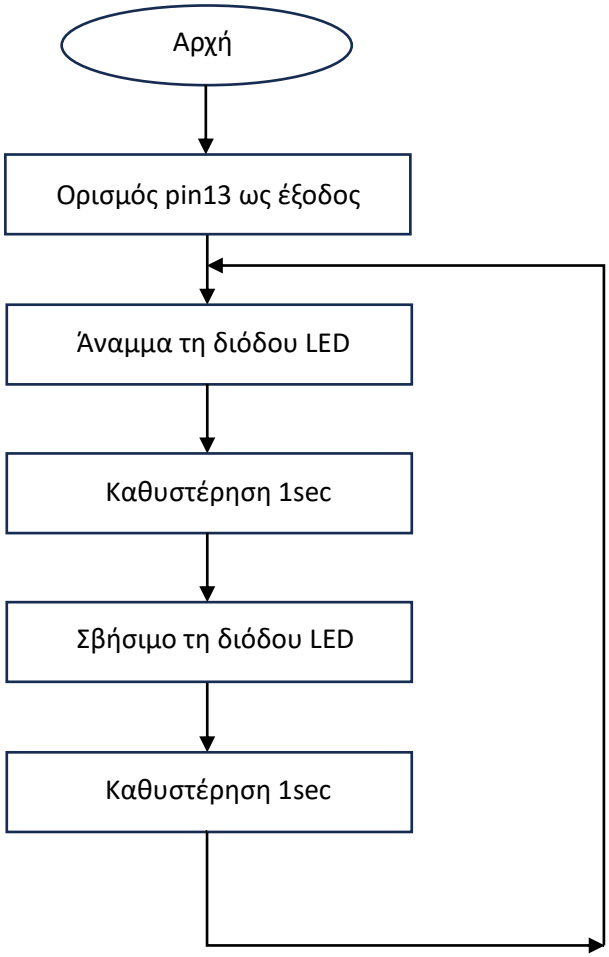
Αν φορτώσουμε το παραπάνω πρόγραμμα στον μικροελεγκτή της πλακέτας Arduino UNO, θα παρατηρήσουμε ότι η δίοδος LED παραμένει μόνιμα αναμένει. Αν είμαστε περισσότερο παρατηρητικοί θα καταλάβουμε ότι η φωτεινότητα της διόδου είναι μειωμένη. Αυτό συμβαίνει διότι, στην πραγματικότητα η δίοδος πράγματι αναβοσβήνει, όμως αναβοσβήνει πολύ γρήγορα[2] με αποτέλεσμα να μη γίνεται αντιληπτή από το ανθρώπινο μάτι (γίνεται όμως αντιληπτή η διαφορετικότητα και συγκεκριμένα η μείωση της φωτεινότητας).

Για να μπορέσουμε να αντιληφθούμε την εναλλαγή τη διόδου LED από ανοικτή σε κλειστή (ON/OFF) και αντίστροφα, θα χρησιμοποιήσουμε μία εντολή καθυστέρησης του μικροελεγκτή. Η εντολή αυτή ονομάζεται **delay()**, αποτελεί συνάρτηση του Arduino και λαμβάνει ένα μόνο όρισμα. Το όρισμα καθορίζει το χρόνο καθυστέρησης σε msec (δηλαδή σε χιλιοστά του δευτερολέπτου).

Το τελικό πρόγραμμα της **Εικόνας 7** καθυστερεί ένα sec τη λειτουργία του μικροελεγκτή στην κατάσταση ON της διόδου (γραμμή 7), και ένα sec τη λειτουργία του μικροελεγκτή στην κατάσταση OFF της διόδου (γραμμή 9). Θα πρέπει να σημειωθεί ότι, οι εντολές των γραμμών 6-9 επαναλαμβάνονται ατέρμονα, όπως γνωστοποιεί και το όνομα της συνάρτησης **loop()**.

[2] Χονδρικά θα μπορούσαμε να πούμε ότι η εκτέλεση εντολών από τον μικροελεγκτή Arduino UNO γίνεται περίπου σε μsec, δηλαδή σε εκατομμυριοστά του sec.

Εικόνα 7 Αναβόσβησμα LED με καθυστέρηση σε κάθε κατάσταση (ON/OFF) 1sec



Εικόνα 8 Διάγραμμα ροής του παραδείγματος

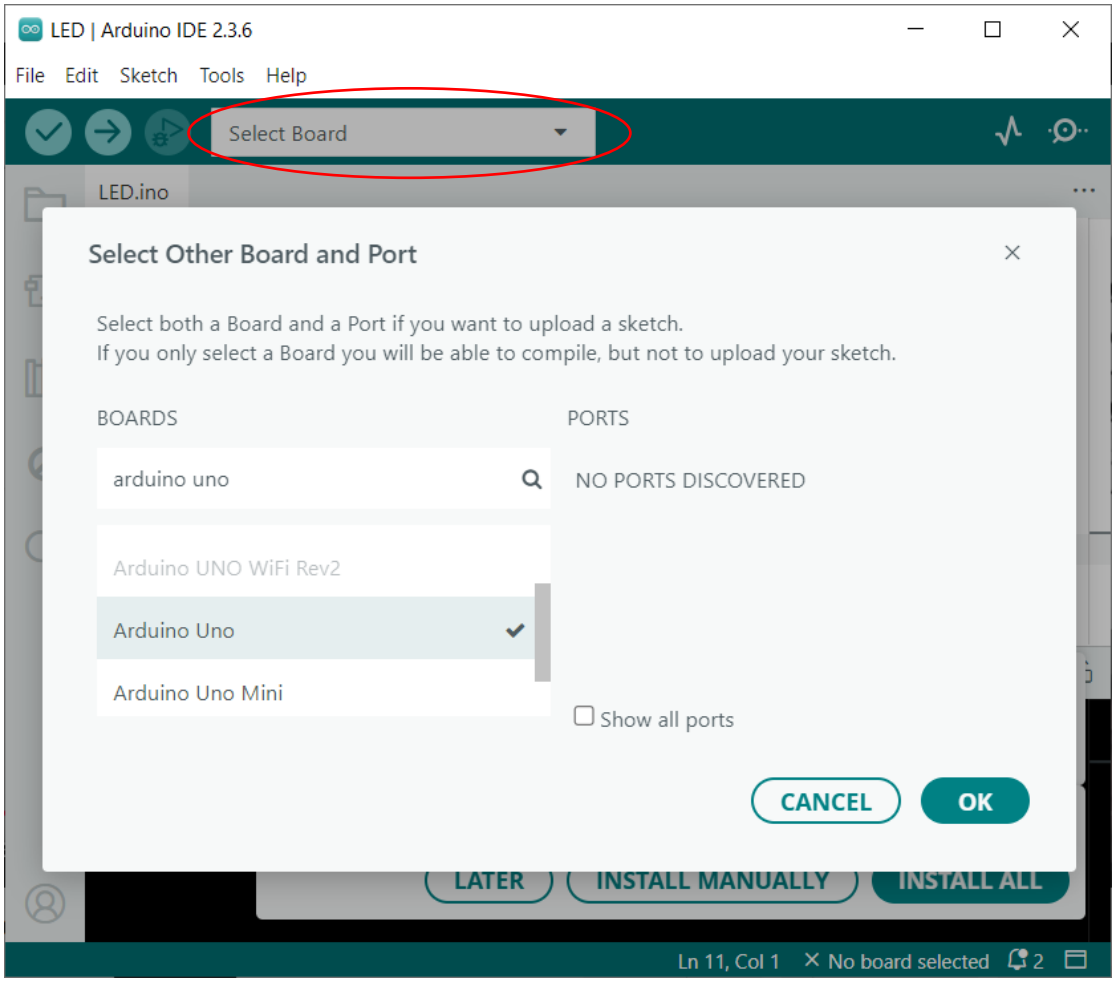
Όλοι οι συμβατικοί υπολογιστές που γνωρίζουμε, πραγματοποιούν αυτό που ονομάζεται *ακολουθιακή εκτέλεση εντολών*. Δηλαδή, ο υπολογιστής εκτελεί ακολουθιακά (σειριακά) την μία εντολή μετά την άλλη. Στο πρόγραμμα μας, οι εντολές είναι οι γραμμές του κώδικα που ολοκληρώνονται με ελληνικό ερωτηματικό (;). Συνεπώς, ο μικροελεγκτής της πλακέτας Arduino UNO εκτελεί πρώτα απ' όλα την εντολή της γραμμής 2, κατόπιν την εντολή της

γραμμής 6, την εντολή της γραμμής 7, έπειτα της 8, και τέλος της γραμμής 9. Επειδή όμως οι εντολές των γραμμών 6-9 βρίσκονται μέσα στην συνάρτηση επανάληψης, με το πέρας της εκτέλεση της εντολής 9, η ροή του κώδικα μεταβαίνει και πάλι στην εντολή της γραμμή 6. Έτσι, επαναλαμβάνεται για πάντα η εκτέλεση των γραμμών 6-9 που αναγκάζουν τη δίοδο LED να αναβοσβήνει με καθυστέρηση 1 sec σε κάθε κατάσταση (ON/OFF).

Στον προγραμματισμό συχνά χρησιμοποιούνται εικόνες αναπαράστασης του κώδικα για διευκόλυνση δημιουργίας ενός νέου ή κατανόησης ενός υπάρχοντος κώδικα. Μία τέτοια μορφή αναπαράσταση είναι και αυτή της **Εικόνας 8**. Οι μορφές αυτές αναπαράστασης ενός κώδικα αυτά ονομάζονται *διαγράμματα ροής*.

Βήματα προγραμματισμού της πλακέτας Arduino UNO

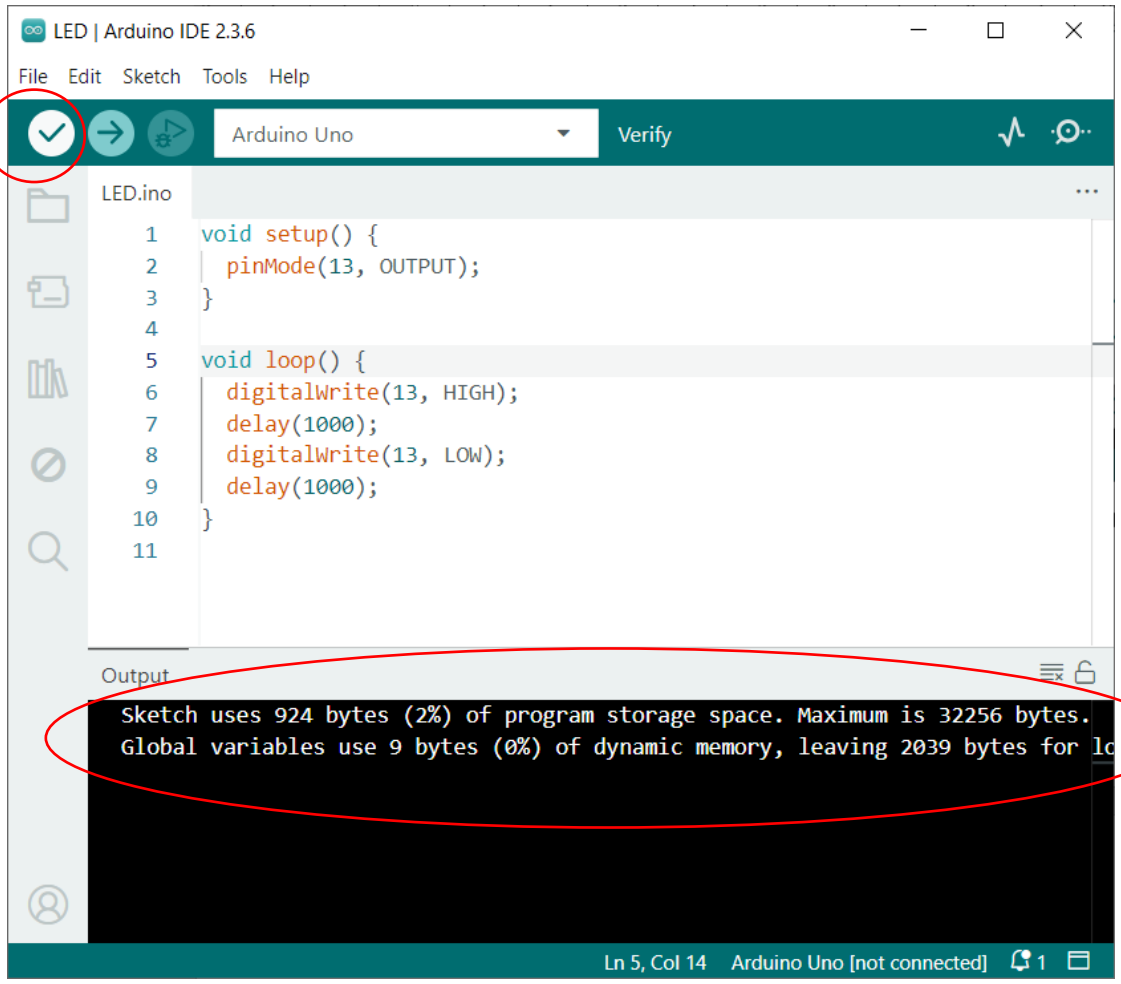
Προτού μεταγλωττίσουμε τον κώδικα που γράψαμε σε γλώσσα μηχανής και φορτώσουμε το αρχείο στην πλακέτα μας, θα πρέπει να ορίσουμε τον τύπο της πλακέτας μικροελεγκτή που χρησιμοποιούμε (στη συγκεκριμένη περίπτωση την πλακέτα Arduino UNO). Η επιλογή γίνεται κάνοντας αριστερό κλικ στο πεδίο Select Board (**Εικόνα 9**).



Εικόνα 9 Επιλογή πλακέτας από το Arduino IDE

Για να ελέγξουμε την ορθότητα του κώδικά μας, όσον αφορά συντακτικά λάθη, κάνουμε αριστερό κλικ στο κουμπί **Verify (√)** που βρίσκεται πάνω-αριστερά στην πράσινη λεζάντα

(**Εικόνα 10**). Αν ο κώδικας είναι σωστός, τότε μας εμφανίζει πληροφορίες σχετικά με το χώρο που δεσμεύει το πρόγραμμά μας στη μνήμη του μικροελεγκτή (κάτω μέρος της **Εικόνας 10**).



Εικόνα 10 Επιλογή πλακέτας από το Arduino IDE

Αν υπάρχει συντακτικό λάθος στον κώδικα τότε στο κάτω μέρος του Arduino IDE θα μας εμφανίσει τα λάθη, ενώ παράλληλα τονίζει με χρωματιστή λεζάντα την πρώτη γραμμή κώδικα στην οποία εντοπίζεται το λάθος.

Για τον προγραμματισμό της πλακέτα και αφού έχει συνδεθεί η πλακέτα στη θύρα USB του προσωπικού μας υπολογιστή, θα πρέπει να κάνουμε αριστερό κλικ στο βελάκι που βρίσκεται δίπλα από το πλήκτρο **Verify (√)**, και το οποίο ονομάζεται **Upload (→)**. Κατά τον προγραμματισμό της πλακέτας γίνεται πρώτα μεταγλώττιση του κώδικα και τη στιγμή που «κατεβαίνει» ο κώδικας τη μνήμη του μικροελεγκτή αναβοσβήνουν κάποιες δίοδοι LED που βρίσκονται στην πλακέτα του Arduino UNO. Με το πέρας του προγραμματισμού ξεκινάει η εκτέλεση των εντολών από το Arduino UNO.

ΕΡΓΑΣΙΑ 1: Σήμα SOS με κώδικα MORSE

Με την ίδια συνδεσμολογία δημιουργείστε σήμα SOS του κώδικα MORSE. Ο κώδικας Morse μεταδίδει πληροφορίες με παλμούς μικρής και μεγάλης διάρκειας (τελείες και παύλες). Το γράμμα S αποτελείται από τρεις συνεχόμενες τελείες, ενώ το γράμμα O από τρεις συνεχόμενες παύλες.

Λύση:

Ex02_SOS.ino

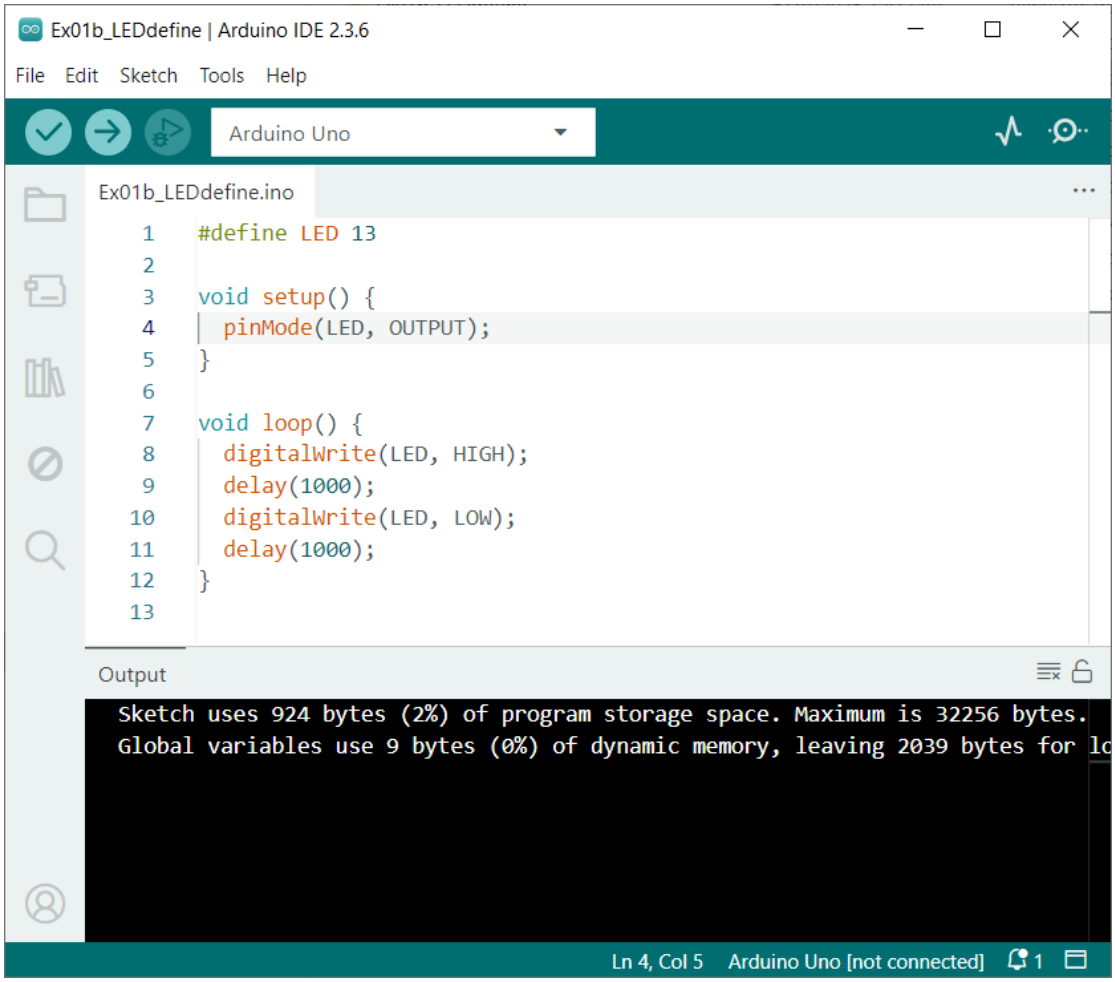
```
void setup() {

  pinMode(13,OUTPUT);

}

void loop() {

  digitalWrite(13,HIGH); //s
  delay(300);
  digitalWrite(13,LOW);
  delay(100);
  digitalWrite(13,HIGH);
  delay(300);
  digitalWrite(13,LOW);
  delay(100);
  digitalWrite(13,HIGH);
  delay(300);
  digitalWrite(13,LOW);
  delay(100);
  delay(1200);

  digitalWrite(13,HIGH); //o
  delay(700);
  digitalWrite(13,LOW);
  delay(100);
  digitalWrite(13,HIGH);
  delay(700);
  digitalWrite(13,LOW);
  delay(100);
  digitalWrite(13,HIGH);
  delay(700);
  digitalWrite(13,LOW);
  delay(100);
  delay(1200);

  digitalWrite(13,HIGH); //s
  delay(300);
  digitalWrite(13,LOW);
  delay(100);
  digitalWrite(13,HIGH);
  delay(300);
  digitalWrite(13,LOW);
  delay(100);
  digitalWrite(13,HIGH);
  delay(300);
  digitalWrite(13,LOW);
  delay(100);

  delay(2000);
}

```

ΕΡΓΑΣΙΑ 2: Βελτίωση του κώδικα με χρήση ΣΤΑΘΕΡΑΣ

Στον κώδικα της εργασίας 1, είναι ιδιαίτερα χρονοβόρα η διαδικασία αλλαγής της διόδου LED και η σύνδεσή της σε διαφορετικό pin του μικροελεγκτή (π.χ. στο pin12). Για το λόγο αυτό μπορούμε να χρησιμοποιήσουμε ένα συμβολικό όνομα για το pin που οδηγεί τη δίοδο LED μέσω τη οδηγίας #define. Η οδηγία αυτή ορίζει μία ΣΤΑΘΕΡΑ του κώδικα, όπως ονομάζεται, και συντάσσεται στην πρώτη γραμμή του προγράμματος. Για παράδειγμα, η ακόλουθη οδηγία ορίζει το συμβολικό όνομα LED για το pin13. Συνεπώς, μπορούμε να αντικαταστήσουμε όλα τα ορίσματα των εντολών του προγράμματός μας που χρησιμοποιούν το pin13, και αντί για τον αριθμό 13 να συντάξουμε το συμβολικό όνομα LED. Με αυτόν τον τρόπο αν θελήσουμε αργότερα να αλλάξουμε θέση στην δίοδο LED και να την οδηγούμε από το pin12 για παράδειγμα, αρκεί να αλλάξουμε την πρώτη γραμμή του κώδικά μας (δηλαδή την οδηγία #define)

#define LED 13

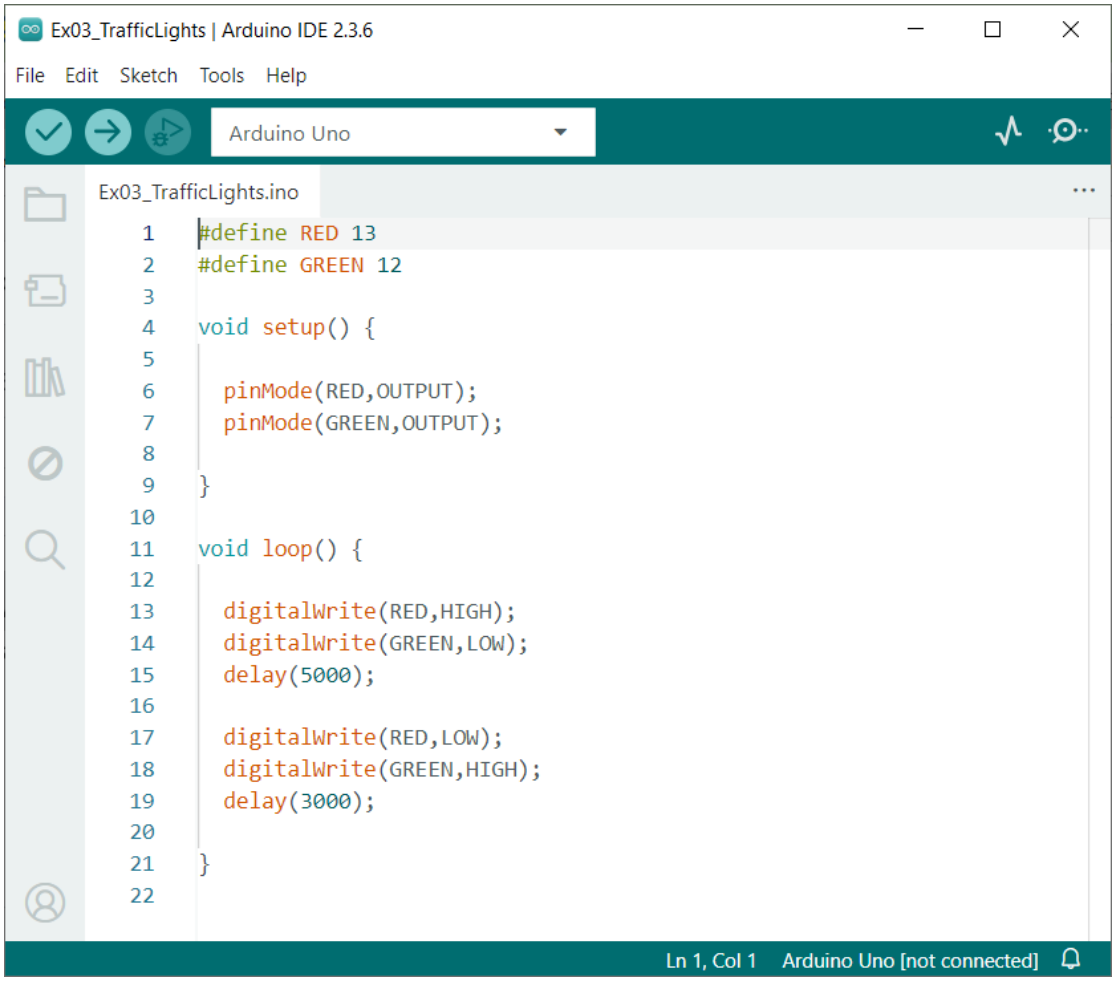
Για παράδειγμα, το πρόγραμμα που αναβοσβήνει το LED ανά 1sec γίνεται ως εξής:



ΕΡΓΑΣΙΑ 3: Φανάρια πεζών

Υλοποιείστε φανάρι πεζών χρησιμοποιώντας δυο διόδους LED οι οποίες θα συνδέονται στο pin13 (κόκκινο φανάρι) και στο pin12 (πράσινο φανάρι).

Λύση:



ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΣΠΙΤΙ: Φανάρια αυτοκινήτων-πεζών

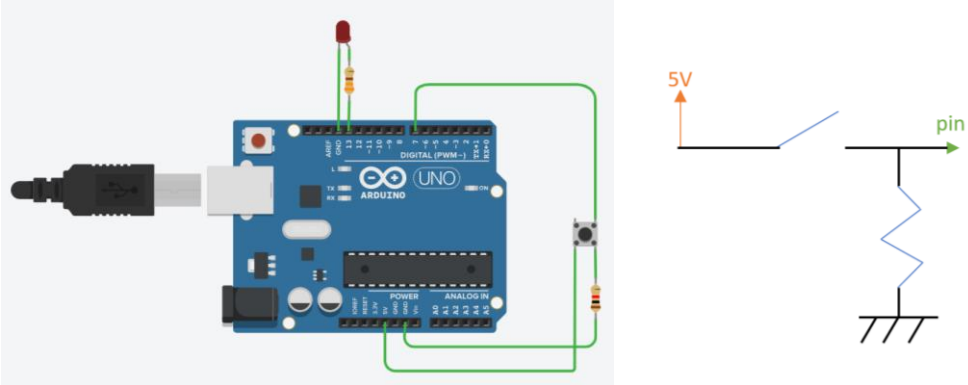
Υλοποιείστε φανάρι αυτοκινήτων και πεζών χρησιμοποιώντας τρεις διόδους LED (με χρώμα κόκκινο, πορτοκαλί, πράσινο) για τα αυτοκίνητα και δυο διόδους για του πεζούς διόδους (με χρώμα κόκκινο, πράσινο).

ΕΡΓΑΣΙΑ ΣΤΗΝ ΤΑΞΗ: Το περιβάλλον Tinkercad

Δημιουργία κυκλωμάτων με το λογισμικό Tinkercad

2η εφαρμογή με Arduino UNO: σύνδεση και ανάγνωση διακόπτη

Στη συνέχεια θα παρουσιάσουμε κώδικα σε Arduino UNO, ο οποίος αναγνωρίζει αν ένας διακόπτης είναι ανοικτός ή κλειστός. Συγκεκριμένα η εφαρμογή είναι μία παραλλαγή-βελτίωση του κώδικα SOS, η οποία ενεργοποιεί την αποστολή μηνύματος SOS μόνο κατά τη δέσμευση του διακόπτη. Το κύκλωμα σύνδεσης της εφαρμογής παρουσιάζεται την **Εικόνα 11**.



Εικόνα 11 Σύνδεση LED και Push Button στην πλακέτα Arduino UNO

Στην αριστερή πλευρά της **Εικόνα 11** παρουσιάζεται η διασύνδεση ενός διακόπτη τύπου Push Button στον ακροδέκτη 7 του Arduino UNO, ενώ το σχηματικό διάγραμμα της διασύνδεσης απεικονίζεται το δεξιό μέρος της εικόνας. Σύμφωνα με το σχηματικο διάγραμμα, όταν ο διακοπτης είναι ανοικτός (όπως ακριβώς φαίνεται στην εικόνα), ο ακροδέκτης (pin) του μικροελεγκτή οδηγείται μέσω της αντίστασης στην γείωση (GND) της πλακέτας και συνεπώς, το pin του μικροελεγκτή αναγνωρίζει λογικό «0». Όταν κλείσει ο διακόπττης, το pin του μικροελεγκτή οδηγείται σε τάση 5V και έτσι, ο μικροελεγκτής λαμβάνει λογικό «1». Η αντίσταση είναι απαραίτητη ώστε να μη δημιουργηθεί βραχυκύκλωμα μεταξύ τάσης (5V) και γείωσης (0V), ενώ μία τυπική τιμή αντίστασης είναι τα 10 kilohm (KΩ).

Η δομή ελέγχου (ροής του κώδικα) do while()

Για την αναγνώριση της λογικής τιμή «0» και «1» στο πρόγραμμα Arduino (που προκύπτει αντιστοίχως όταν ο διακόπητης είναι ανοικτός και κλειστός) θα χρησιμοποιηθεί μία δομή ελέγχου της ακολουθιακής ροής του προγράμματος. Η δομή αυτή ελέγχου είναι η **do while()**, η οποία υλοποιεί ένα βρόχο επανάληψης σε περίπτωση που επαληθεύεται μία συνθήκη. Σε αντίθεση με την δομή επανάληψης **loop()**, η οποία εκτελεί ατέρμονα τις εντολές που εσωκλείονται στα άγκυστρα (**{}**) της δομής, η **do while()** εκτελεί τις εντολές που εσωκλείονται στα άγκυστρα (**{}**) της δομής, σε περίπτωση όμως που επαληθεύεται η συνθήκη ελέγχου αυτής. Η τελευταία τοποθετείται εντός παρανθέσεων ακριβώς μετά την κωδική λέξη while.

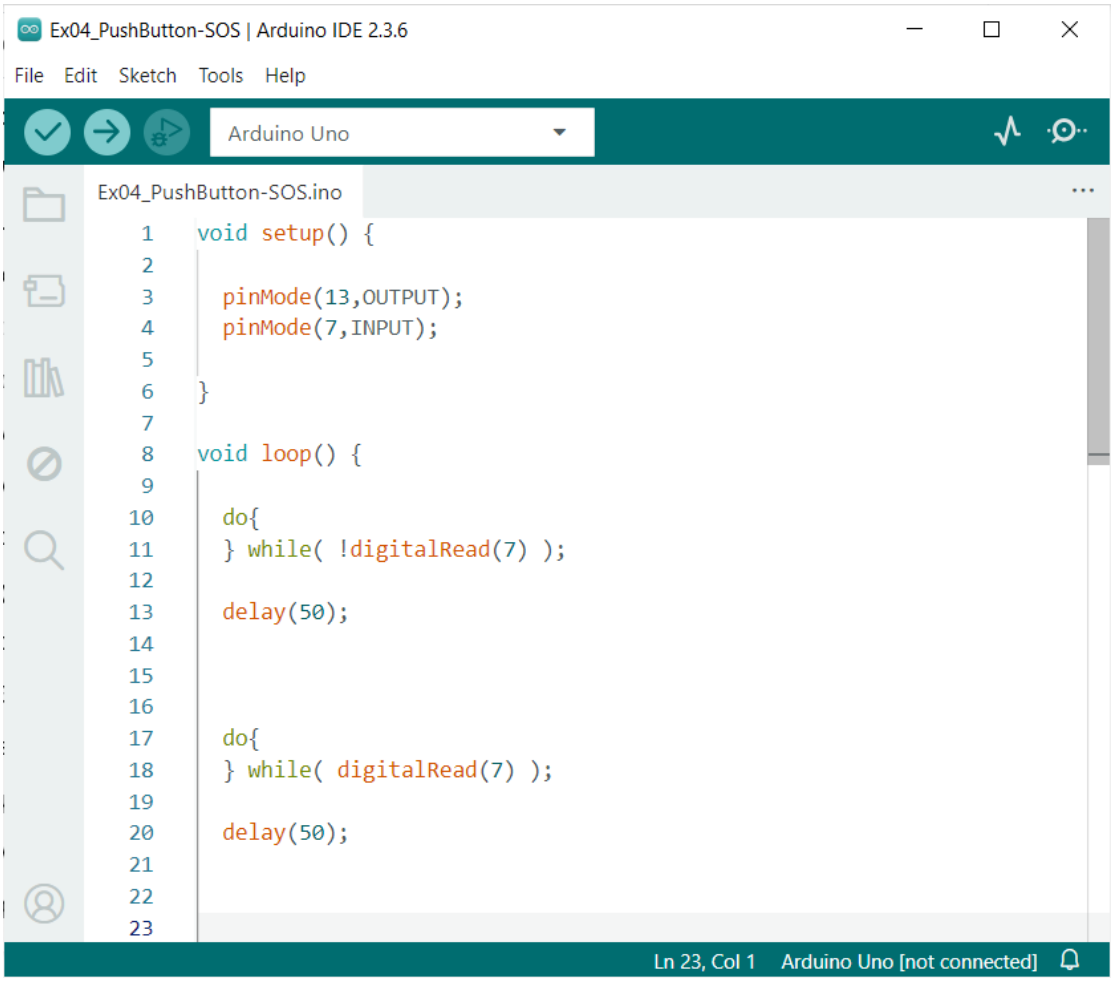
```
do{
} while( !digitalRead(7) );
```

```
do{
} while( digitalRead(7) );
```

Εικόνα 12 Η δομή ελέγχου **do while()** για την αναγνώριση της κατάστασης του διακόπτη

Στην **Εικόνα 12** παρουσιάζεται ο έλεγχος ανοικτού/κλειστού διακόπτη με τη χρήση της **do while()**. Η συνάρτηση **digitalRead()** που τοποθετείται εντός των παρενθέσεων (και αποτελεί της συνθήκη ελέγχου της δομής), χρησιμοποιείται για να διαβάσει (Read) την ψηφιακή (digital) τιμή που έχει ο διακόπτης εισόδου, στη συγκεκριμένη περίπτωση τη λογική κατάσταση (1 ή 0) στην οποία βρίσκεται το pin7. Σύμφωνα με όσα αναφέραμε πρωτύτερα, όταν εκτελείται η **digitalRead()** επιστρέφει λογικό «1» ή λογικό «0». Επομένως η συνθήκη της

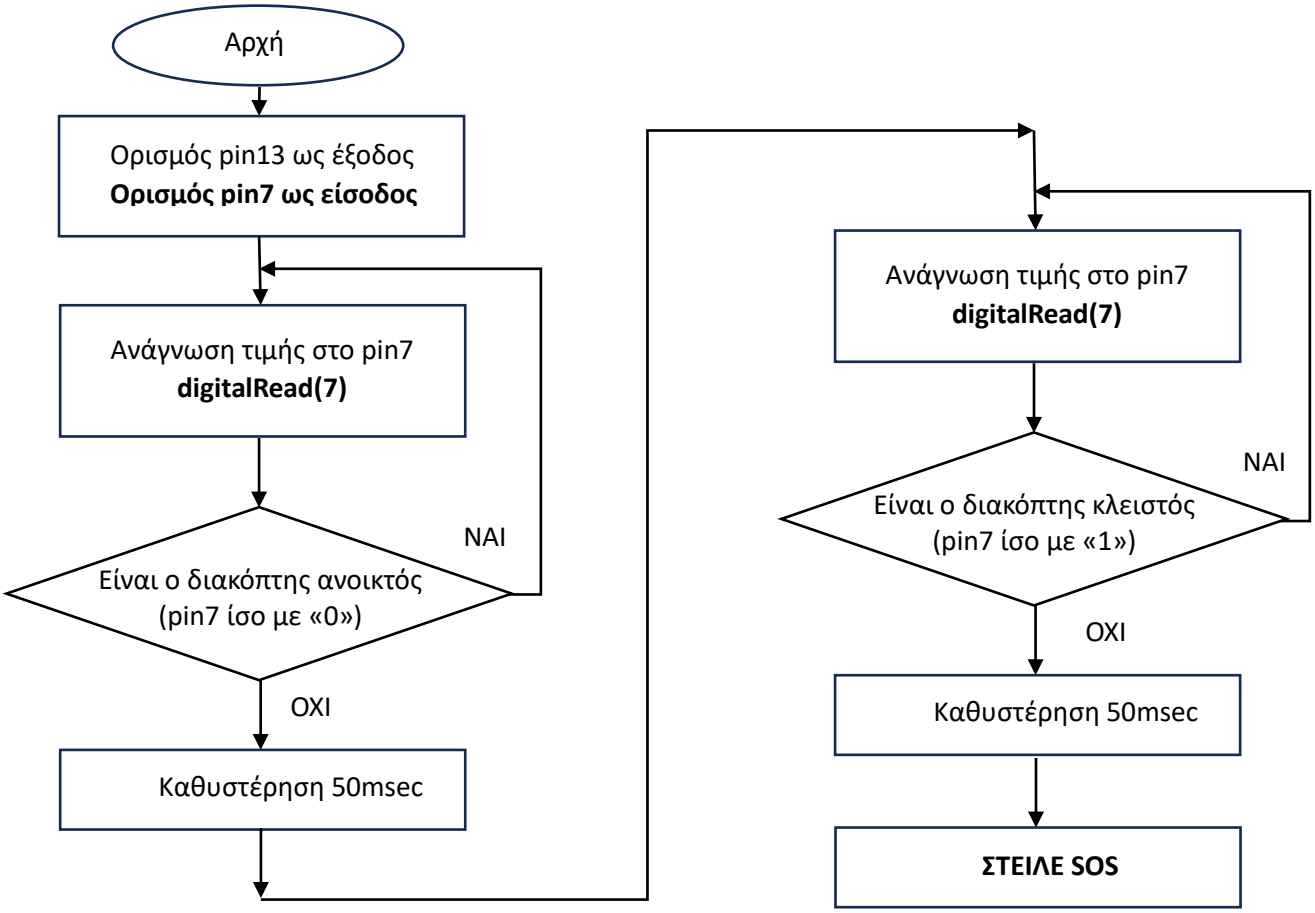
do while() ορίζει την επανάληψη των εντολών αυτής (εντός των αγκίστρων) όταν η συνθήκη ελέγχου είναι αληθής. Ο όρος αληθής ταυτίζεται με την κατάσταση λογικό «1», όπως περιγράφεται από το παράδειγμα στην δεξιά πλευρά της **Εικόνα 12**, η οποία επαληθεύει ότι ο διακόπτης είναι κλειστός. Στην περίπτωση που θέλουμε να ελέγξουμε ότι ο διακόπτης είναι ανοικτός, τότε τοποθετούμε ένα θαυμαστικό (!) μπροστά από την **digitalRead()**. Η σύνταξη **!digitalRead()** ελέγχει την αντίθετη περίπτωση, δηλαδή επαναλαμβάνεται το σώμα της **do while()** στην περίπτωση που η τιμή του pin δεν βρίσκεται σε κατάσταση λογικού «1», άρα σε περίπτωση που βρίσκεται σε λογικό «0».



Εικόνα 13 Κώδικας αναγνώρισης της ψηφιακής τιμής («1» ή «0») ενός διακόπτη

Στην **Εικόνα 13** παρουσιάζεται ο κώδικας που ορίζει τον pin 7 ως θύρα εισόδους (γραμμή 4) και κατόπιν ελέγχει την κατάσταση του διακόπτη εντός του loop(). Συγκεκριμένα, η συνθήκη ελέγχου στην γραμμή 11 επαναλαμβάνει την εκτέλεση της do while() από τη γραμμή 10 στην περίπτωση που ο διακόπτης παραμένει ανοικτός (στην περίπτωση δηλαδή που η τιμή του pin δεν είναι «1», άρα στην περίπτωση που το pin βρίσκεται σε λογικό «0»). Συνεπώς το πρόγραμμα περιμένει αρχικά να πατηθεί ο διακόπτης. Όταν πατηθεί ο διακόπτης συνεχίζει η ακολουθιακή εκτέλεση των εντολών, και αφού εκτελεστεί η καθυστέρηση των 50msec της γραμμής 13, εκτελείται η do while των εντολών 17, 18. Η συνθήκη της do while στη γραμμή 18 ελέγχει την κατάσταση στην οποία βρίσκεται το pin7, και αν η τιμή αυτού είναι λογικό «1» (δηλαδή ο διακόπτης παραμένει πατημένος) τότε επαναλαμβάνεται η do while() από τη γραμμή 17.

Στο σημείο αυτό αξίζει να σημειωθεί ότι το σώμα της do while() και στις 2 περιπτώσεις δεν περιλαμβάνει κάποιες εντολές, αφού σκοπός αυτών είναι να «παγώνουν» την ακολουθιακή ροή εκτέλεσης εντολών μέχρι να αλλάξει κατάσταση ο διακόπτης, όπως ακριβώς περιγράφεται στο διάγραμμα ροής της **Εικόνας 14**.



Εικόνα 14 Διάγραμμα ροής του παραδείγματος SOS με εκκίνηση από push-button

Σύμφωνα με το διάγραμμα ροής, το πρόγραμμα περιμένει να πατηθεί ο διακόπτης, στην συνέχεια το πρόγραμμα περιμένει ξανά έως ότου ελευθερωθεί ο διακόπτης, και τελικά αποστέλλει μήνυμα SOS (με κώδικα μορς). Το κομμάτι του κώδικα που αποστέλλει το SOS είναι ίδιο με το αρχικό παράδειγμα που υλοποιήσαμε πρωτύτερα.

Φαινόμενο αναπηδήσεων στους μηχανικούς διακόπτες

Η καθυστέρηση των 50msec ακριβώς μετά την αλλαγή της κατάστασης στον διακόπτη (από ανοικτό σε κλειστό και αντίστροφα) απαιτείται διότι, όλοι οι μηχανικοί διακόπτες παρουσιάζουν το λεγόμενο *φαινόμενο αναπηδήσεων (bouncing)*. Όπως μία μπάλα μπάσκετ θα αναπηδήσει μερικές φορές στο έδαφος όταν ελευθερωθεί από τα χέρια μας, έως ότου σταματήσει τελείως, έτσι και οι μηχανικοί διακόπτες τη στιγμή που δεσμεύονται από τον χρήστη, ανοιγοκλείνουν για ένα μικρό χρονικό διάστημα έως ότου σταθεροποιηθούν στην κατάσταση *κλειστού διακόπτη*. Το ίδιο συμβαίνει και όταν ο χρήστης ελευθερώνει το

διακόπτη, δηλαδή και πάλι ανοιγοκλείνει ο διακόπτης έως ότου σταθεροποιηθεί στην κατάσταση *ανοικτού διακόπτη*. Για το λόγο αυτό, στα ψηφιακά κυκλώματα χρησιμοποιείται μία καθυστέρηση χρόνου ακριβώς μετά την ανίχνευση αλλαγής στην κατάσταση του διακόπτη, ώστε να αποσβέσουν οι αναπηδήσεις.

ΕΡΓΑΣΙΑ ΣΤΗΝ ΤΑΞΗ: Δημιουργία νέων συναρτήσεων

Μελετήστε τον κώδικα μορς και τροποποιείστε τη δομή loop() ώστε το κάθε γράμμα που θέλετε να αποστείλετε να δημιουργείται με την κλήση δύο συναρτήσεων α) τη συνάρτηση dot() που θα αποστέλλει τελεία μαζί το επακόλουθο κενό και β) τη συνάρτηση dash() που θα αποστέλλει παύλα (επίσης μαζί με το επακόλουθο κενό).

Υπόδειξη: τοποθετείστε τις συναρτήσεις στην αρχή του κώδικα (μετά τη δήλωση τυχών σταθερών), ενώ οι συναρτήσεις που θα δημιουργήσετε δε θα έχουν ορίσματα και δε θα επιστρέφουν κάποια τιμή (άρα θα είναι συναρτήσεις τύπου void). Παρακάτω δίδεται ως παράδειγμα η συνάρτηση dot():

```
void dot (){
  digitalWrite (LED, HIGH);
  delay(300);
  digitalWrite(LED, LOW);
  delay(100);
}
```

ΕΡΓΑΣΙΑ ΣΤΗΝ ΤΑΞΗ: Αποστολή μηνύματος SOS στον Η/Υ

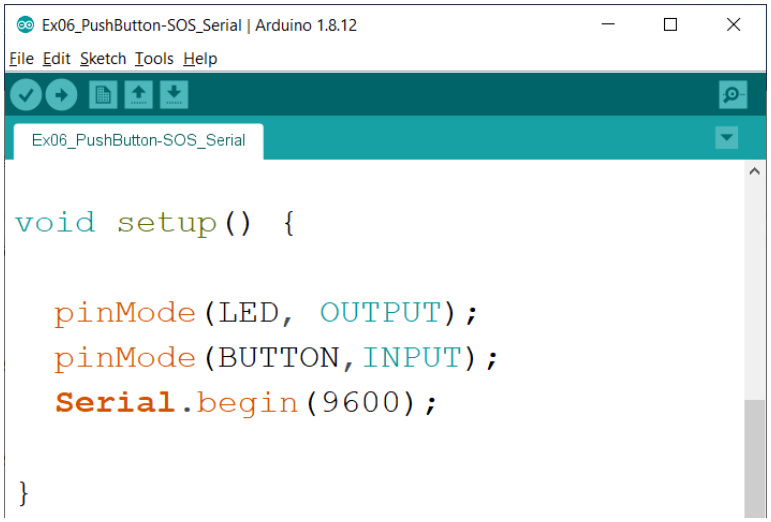
Η πλακέτα Arduino έχει τη δυνατότητα να αποστέλλει και να λαμβάνει μηνύματα προς/από τη θύρα USB, μέσω ειδικών συναρτήσεων που έχουν δημιουργηθεί, καθώς και της εφαρμογής Serial Monitor που είναι ενσωματωμένη στο περιβάλλον Arduino IDE (διαθέσιμη από το μενού Tools → Serial Monitor). Η ενεργοποίηση της επικοινωνίας με τον Η/Υ γίνεται μέσω της συνάρτησης **Serial.begin()**, η οποία συντάσσεται εντός της setup() και λαμβάνει ως όρισμα το ρυθμό μετάδοσης σειριακών δεδομένων (baud rate). Οι τιμές baud είναι συγκεκριμένες (π.χ. 9600, 19200, κλπ.) και η τιμή που ορίζεται στη **Serial.begin()** θα πρέπει να συμφωνεί με την τιμή που ορίζεται (από το χρήστη) στο παράθυρο της εφαρμογής Serial Monitor.

Ως άσκηση στην τάξη στείλτε μήνυμα στη σειριακή θύρα USB του Η/Υ τοποθετώντας την παρακάτω εντολή στην κατάλληλη θέση μέσα στο βρόχο loop():

Serial.print("SOS");

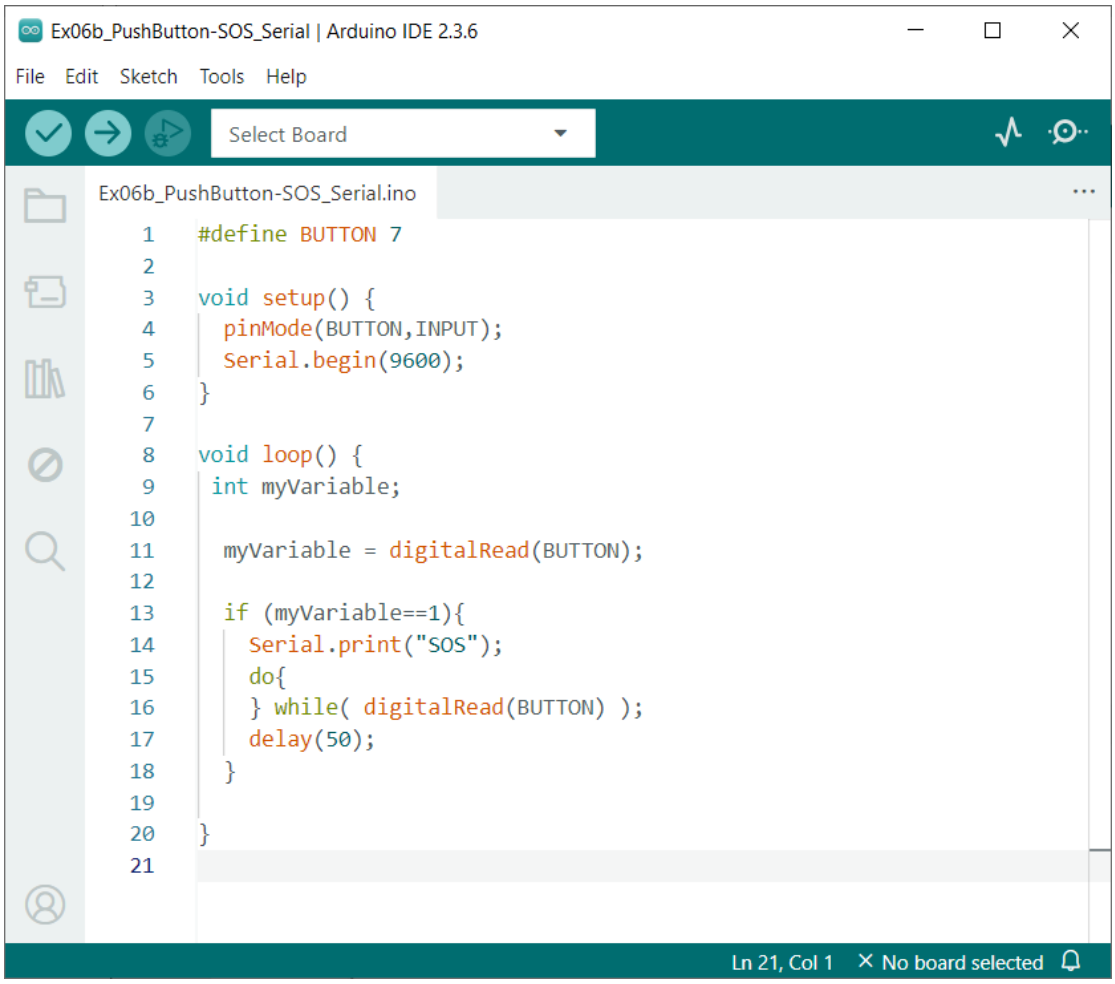
Επιπροσθέτως, παρατηρείστε την αλλαγή δεδομένων που αποστέλλονται με τη χρήση της συνάρτησης:

Serial.println("SOS");



ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΣΠΙΤΙ: Αποστολή μηνύματος SOS με χρήση **if()**

Το ακόλουθο πρόγραμμα χρησιμοποιεί μία δομή ελέγχου της ακολουθιακής ροής του κώδικα, τη δομή if(), το σώμα της οποίας εκτελείται όταν επαληθεύεται ως αληθής η συνθήκη της δομής (η οποία βρίσκεται εντός των παρενθέσεων αυτής).



Στο συγκεκριμένο πρόγραμμα γίνεται χρήση μιας μεταβλητής, που ορίζεται εντός του loop (γραμμή 9) και όπως γνωστοποιεί το όνομά της, έχει τη δυνατότητα να αλλάζει το περιεχόμενό της κατά τη διάρκεια εκτέλεσης του κώδικα. Κατά συνέπεια, κάθε φορά που εκτελείται ο βρόχος loop() η μεταβλητή με το όνομα *myVariable* λαμβάνει νέα τιμή, η οποία τιμή είναι η λογική κατάσταση στην οποία βρίσκεται ο διακόπτης (που στο παράδειγμά μας συνδέεται στο pin7). Έτσι, όταν ο διακόπτης είναι ανοικτός η μεταβλητή λαμβάνει τιμή μηδέν (λογικό «0»), ενώ όταν ο χρήστης πατήσει το διακόπτη, η μεταβλητή λαμβάνει τιμή ένα (λογικό «1»).

Η συνθήκη της δομής if() ελέγχει αν έχει πατηθεί ο διακόπτης, χρησιμοποιώντας το σύμβολο της ισότητας (==) για τον κώδικα Arduino, και σε περίπτωση που επαληθεύεται η συνθήκη (δηλαδή αν η μεταβλητή φέρει την τιμή 1) τότε εκτελείται το σώμα της δομής και αποστέλλεται μήνυμα SOS στον Η/Υ.

Ερώτηση: Ποιος είναι ο ρόλος της do while() δομής στο πρόγραμμα; Αφαιρέστε τη δομή (μαζί με την καθυστέρηση των 50msec) και εξηγήστε τις αλλαγές που παρατηρείται κατά την εκτέλεση του προγράμματος. Εξηγείστε τι ακριβώς συμβαίνει.

ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΣΠΙΤΙ: Αποστολή πολλαπλών μηνυμάτων SOS με for()

Αναζητείστε πληροφορίες στο διαδίκτυο για τη σύνταξη και τη χρήση της δομής ελέγχου ροής **for()** στον προγραμματισμό Arduino. Τροποποιείστε το προηγούμενο πρόγραμμα ώστε κατά τη δέσμευση του διακόπτη να αποστέλλονται τρία διαδοχικά μηνύματα SOS με τη χρήση της δομής for().