



ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
DEMOCRITUS UNIVERSITY OF THRACE



ΜΑΘΗΜΑ
ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ (206ΕΥΥΚ)
ΠΠΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΑΡΙΝΟ 2023-2024

Διάλεξη Νο 8:

Εισαγωγή στους μικροελεγκτές RISC AVR - Arduino (Μέρος II)

Δ. Καραμπατζάκης, Επίκουρος Καθηγητής

email. dkara@cs.duth.gr



Πληροφορίες για την επικοινωνία με τον διδάσκοντα

Διδάσκων:

Δημήτρης Καραμπατζάκης, Επίκουρος Καθηγητής

Αναλογικά και Ψηφιακά Ηλεκτρονικά Συστήματα

Μέλος Εργαστηρίου Βιομηχανικών και Εκπαιδευτικών Ενσωματωμένων Συστημάτων

Επικοινωνία:

email. dkara@cs.duth.gr

web. <http://www.internetofthings.gr/>

Ώρες Γραφείου:

μετά από συνεννόηση με email στο γραφείο ΦΕ 315 (πάνω από αιθ. Α1)

Κύριο Σύγγραμμα Μαθήματος (ΕΥΔΟΞΟΣ)

ΟΡΓΑΝΩΣΗ ΚΑΙ ΣΧΕΔΙΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

Το βιβλίο αυτό γράφτηκε για να καλύψει τις ανάγκες των προγραμμάτων σπουδών των Ελληνικών Πανεπιστημίων με ένα σύγγραμμα που θα εισάγει τον φοιτητή στην δομή και τη λειτουργία ενός Υπολογιστικού Συστήματος.

Γίνεται μια προσπάθεια να παρουσιαστούν έννοιες όπως: μικροεπεξεργαστής, μνήμη, αποκωδικοποίηση μνήμης, μέθοδοι διευθυνασιότητας, μονάδες εισόδου-εξόδου, διακοπές, κ.α., οι οποίες όταν κατανοηθούν και γίνουν κτήμα του αναγνώστη θα μπορούσαν να τον βοηθήσουν να κατανοήσει τη δομή και την λογική ενός συστήματος υπολογιστή.

Γίνεται αναφορά στον μικροεπεξεργαστή MC68000 της MOTOROLA, ως «κλασικού εκπροσώπου της τεχνολογίας CISC, στον μικροελεγκτή AVR ATmega8515 της ATMEL, ως «κλασικού εκπροσώπου της τεχνολογίας RISC και στο σύστημα ανάπτυξης εφαρμογών Arduino, που αποτελεί τον πλέον εύχρηστο σύστημα μικροελεγκτή ανοιχτού υλικού και λογισμικού σε εφαρμογές ενσωματωμένων συστημάτων.

Για την παρουσίαση στην αίθουσα, το σύγγραμμα συνοδεύεται από εκατοντάδες διαφάνειες Power Point.

Το υποστηρικτικό υλικό του βιβλίου με τις βιντεοσκοπημένες διαλέξεις, μπορείτε να τις βρείτε αναρτημένες στη σελίδα <https://www.youtube.com/user/dimpogaridis/playlists>.



Ο Δημήτρης Πογαρίδης είναι Διδάκτορας Ηλεκτρονικής και Ηλεκτρολόγος Μηχανικός (B.Sc, M.Sc, Ph.D) του πανεπιστημίου του Salford της Αγγλίας. Υπάρχει επί 40 χρόνια Καθηγητής της Τριτοβάθμιας Εκπαίδευσης στην Ελλάδα με γνωστικό αντικείμενο τα «Ψηφιακά και Μικροηλεκτρονικά Συστήματα».

Το πρώτο του βιβλίο με τίτλο «Μικροηλεκτρονικές» εκδόθηκε το 1994. Από τότε συνέγραψε, μεταξύ άλλων, άλλα οκτώ βιβλία και πλήθος σημειώσεων και βοηθημάτων στο γνωστικό αντικείμενο που πραγματεύονται.

Στην πολυετή ακαδημαϊκή του καριέρα τιμήθηκε με πολλά (16) βραβεία και διακρίσεις για καινοτόμες επιστημονικές δημιουργίες και δράσεις.

Για την εν γένει επιστημονική και κοινωνική του προσφορά του απενεμήθη από το ΥΠΕΠΘ το «Βραβείο Επιστημονικής και Αναπτυξιακής Αριστείας», από τον Δήμαρχο Πτολεμαΐδας (δισέπτη πατρίδα του) η «Κεφαλή του Πτολεμαίου», από τον Δήμο Πτολεμαΐδας, με ομόθυμη απόφαση του Δημοτικού Συμβουλίου, το «Αργυρό Μετάλλιο της Πόλης» και «Ανυμνητικός Πάμπυρος», από τον Νομάρχη Καβάλας «Τιμητική Διάκριση» και από τη Δήμαρχο Καβάλας η «Γκραμούρα του Δήμου Καβάλας».

Από τον ίδιο συγγραφέα κυκλοφορούν μεταξύ άλλων:

- Ψηφιακή Σχεδίαση με τις γλώσσες VHDL και Verilog - Αρχές και Πρακτικές
- Σχεδίαση Συστημάτων Μικροηλεκτρονικών
- Ενσωματωμένα Συστήματα – Οι μικροελεγκτές AVR και ARDUINO

ΠΙ ΔΙΣΙΓΜΑ
ΕΚΔΟΣΕΙΣ

e-mail: info@disigma.gr
www.disigma.gr



ISBN 978-618-5242-61-9



ΠΙ

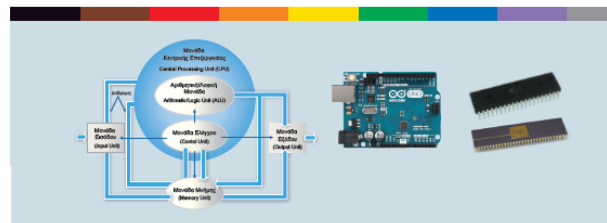
ΠΙ ΔΙΣΙΓΜΑ
ΕΚΔΟΣΕΙΣ

Δημήτρης
Πογαρίδης

ΟΡΓΑΝΩΣΗ ΚΑΙ
ΣΧΕΔΙΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

Δημήτρης Πογαρίδης

ΟΡΓΑΝΩΣΗ ΚΑΙ ΣΧΕΔΙΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ



Οργάνωση και Σχεδίαση Υπολογιστών

Συγγραφέας: Πογαρίδης Δημήτριος

Έτος Έκδοσης: 2019

Κωδικός στον Εύδοξο: 86192986

Λογισμικό - Αναπτυξιακό

- Α' μέρος μαθήματος (CISC):
 - Assembly για τον Motorola68000
 - Λογισμικό easy68k <http://www.easy68k.com/>
- Β' μέρος μαθήματος (RISC):
 - Υλοποίηση σχεδιάσεων σε αναπτυξιακό Arduino (προαιρετική αγορά του σύμφωνα με τις οδηγίες)
 - Λογισμικό Arduino IDE: <https://www.arduino.cc/en/Main/Software>
 - Η γλώσσα προγραμματισμού (C++) και οι εντολές που υποστηρίζει είναι διαθέσιμες στο: <https://www.arduino.cc/reference/en/>

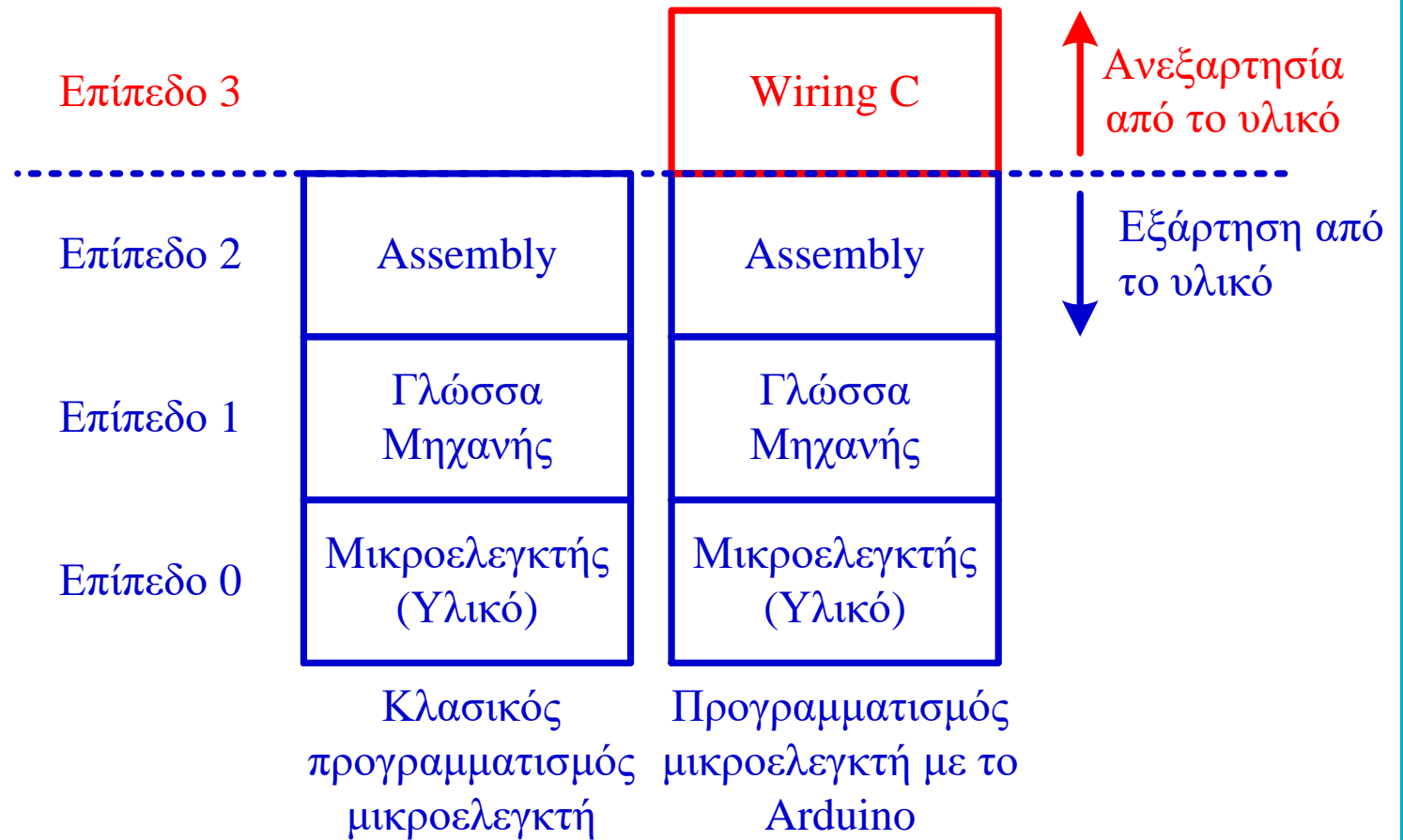
Η ΑΝΑΠΤΥΞΙΑΚΗ ΠΛΑΤΦΟΡΜΑ ARDUINO



Η διαφορά αυτής της πλατφόρμας σε σχέση με τους μικροελεγκτές που έχετε μελετήσει μέχρι τώρα έγκειται στο επίπεδο αφαίρεσης στο οποίο μπορεί να γίνει ο προγραμματισμός.

Με άλλα λόγια, ο προγραμματισμός του μικροελεγκτή στην πλατφόρμα Arduino γίνεται με τη γλώσσα Wiring C που αποτελεί μια παραλλαγή της γνωστής C++.

Έτσι, ο προγραμματισμός μιας τέτοιας πλατφόρμας είναι φιλικότερος και επιτρέπει τη γρηγορότερη ανάπτυξη εφαρμογών χωρίς να προϋποθέτει τη λεπτομερή γνώση της αρχιτεκτονικής του μικροελεγκτή.



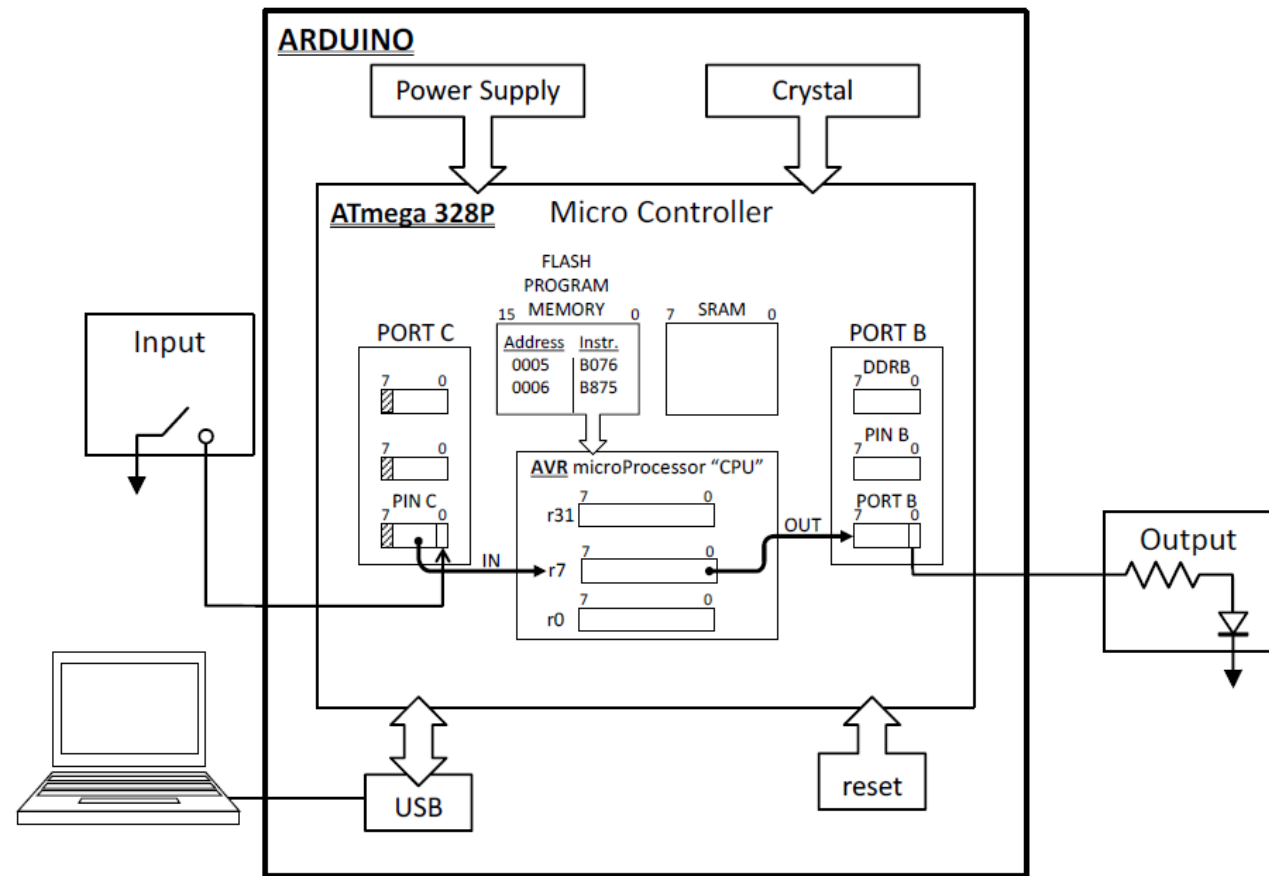
Η ΠΡΟΣΕΓΓΙΣΗ ARDUINO



Ο προγραμματισμός οποιουδήποτε μικροελεγκτή προϋποθέτει την ύπαρξη είτε στοιχειωδών κυκλωμάτων, είτε συγκεκριμένων διασυνδέσεων (μέσω κατάλληλων αντιστάσεων) με ένα υπολογιστικό σύστημα (π.χ. μέσω σειριακής ή παράλληλης θύρας), είτε ειδικού προγραμματιστή.

Στην πλατφόρμα Arduino η συγκεκριμένη τεχνολογία απαλλάσσει το σχεδιαστή ή τον προγραμματιστή από όλες αυτές τις λεπτομέρειες και τα κυκλώματα.

Οτιδήποτε είναι απαραίτητο για τον προγραμματισμό του μικροελεγκτή, βρίσκεται πάνω στην κάρτα στην οποία είναι τοποθετημένος.

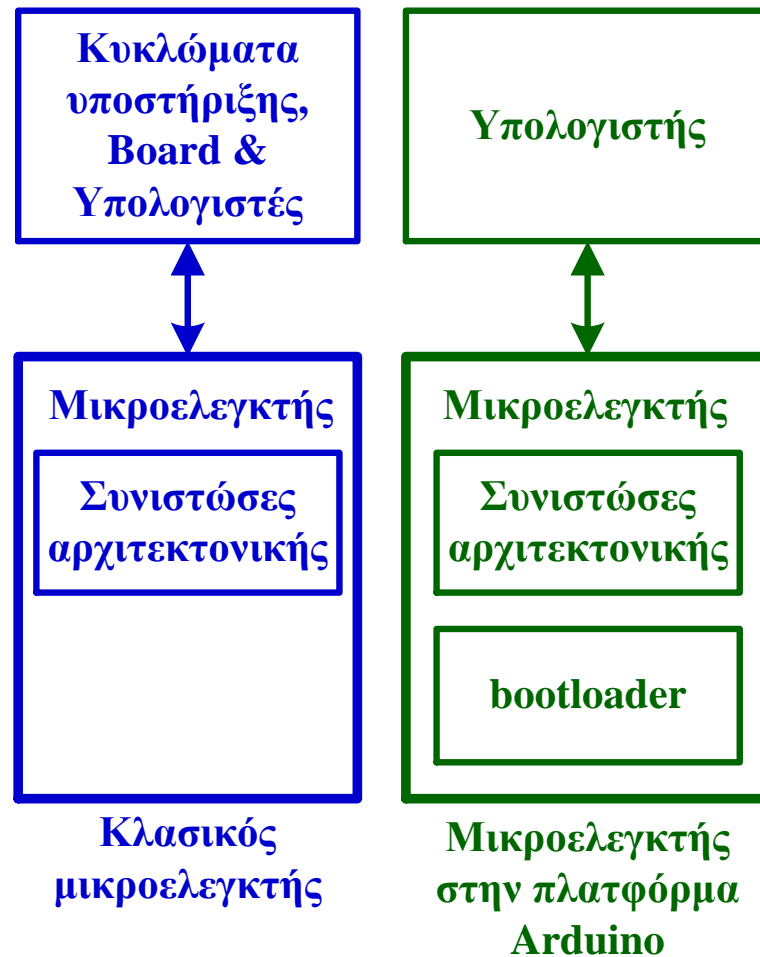


Η ΠΡΟΣΕΓΓΙΣΗ ARDUINO

Το πρόγραμμα αναπτύσσεται στον υπολογιστή, ενώ η μεταφόρτωση του εκτελέσιμου κώδικα γίνεται μέσω της θύρας USB.

Επιπλέον, οι μικροελεγκτές που τοποθετούνται στα Arduino είναι εφοδιασμένοι με ένα ειδικό λογισμικό που ονομάζεται bootloader και δίνει τη δυνατότητα του επαναπρογραμματισμού.

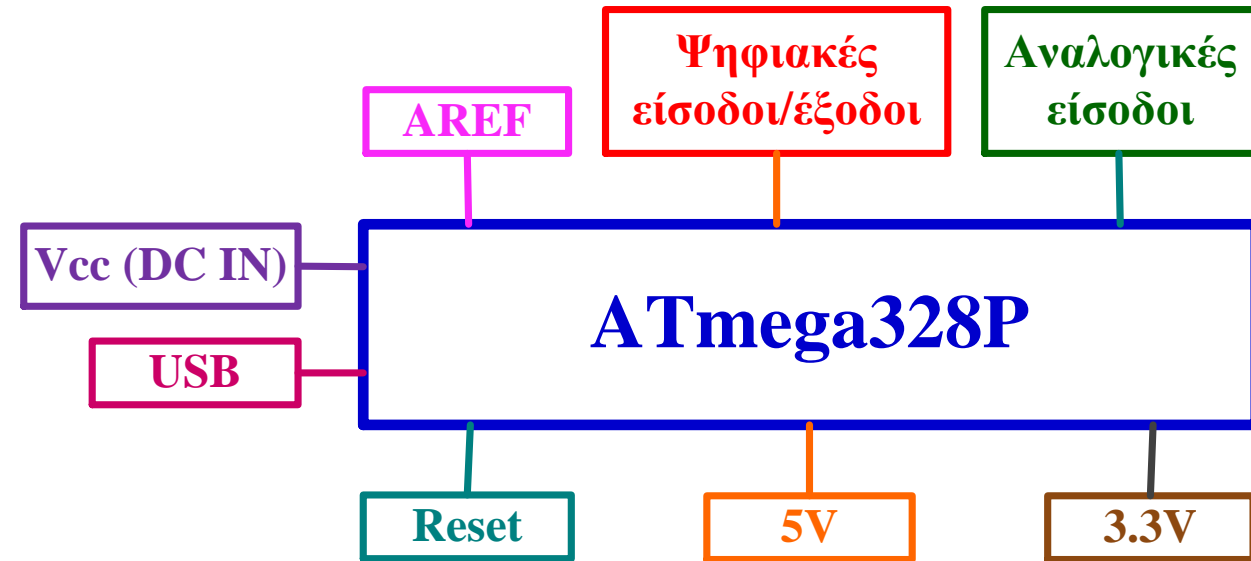
Οι πιο προχωρημένοι έχουν πάντα τη δυνατότητα να προγραμματίσουν τον μικροελεγκτή του Arduino, όπως και κάθε άλλο μικροελεγκτή, αφαιρώντας τον bootloader ή και χρησιμοποιώντας άλλες διασυνδέσεις και κυκλώματα.



ARDUINO UNO (ATmega328P)

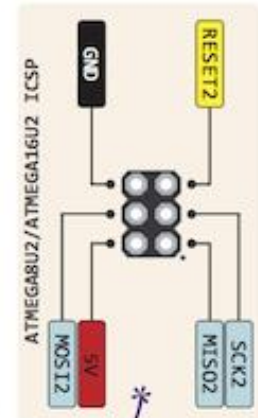
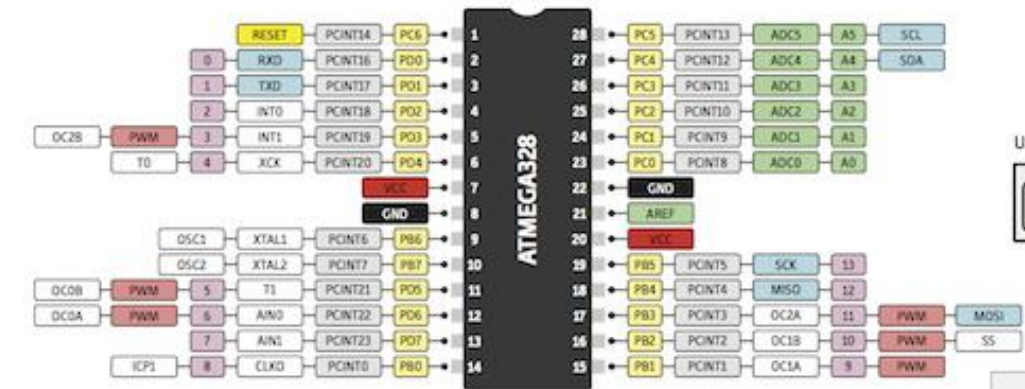


Το ARDUINO UNO



Το Arduino UNO που χρησιμοποιεί τον μικροελεγκτή ATmega328P, είναι η πιο δημοφιλής έκδοση Arduino. Σχεδόν όλες οι υπόλοιπες εκδόσεις βασίζονται στην ίδια φιλοσοφία. Το σχήμα παρουσιάζει σε διάγραμμα βαθμίδας τα βασικά χαρακτηριστικά και δυνατότητες που προσφέρονται μέσω της κάρτας.

THE DEFINITIVE ARDUINO UNO PINOUT DIAGRAM

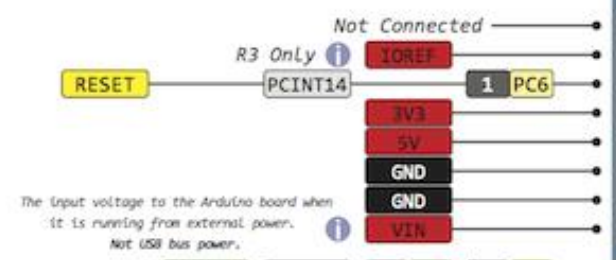


- ⚠ Absolute max per pin 40mA recommended 20mA
- ⚡ Absolute max 200mA for entire package

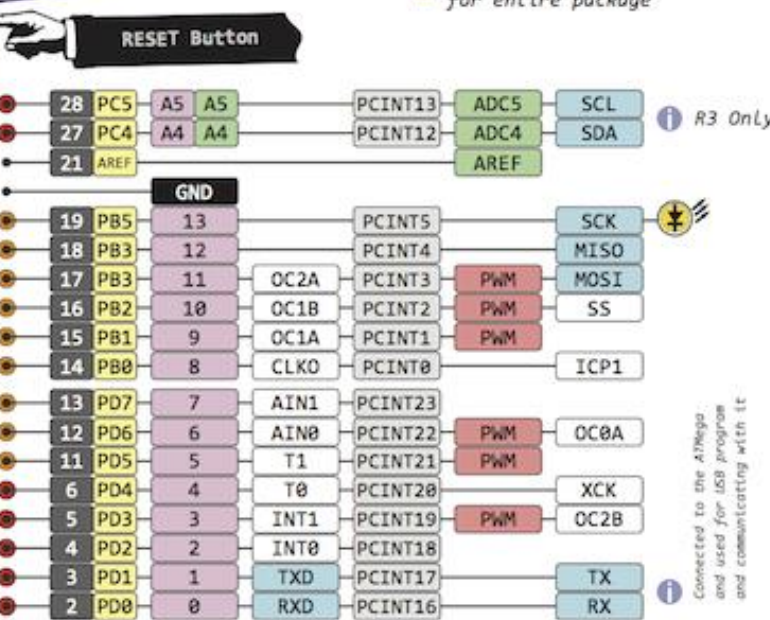
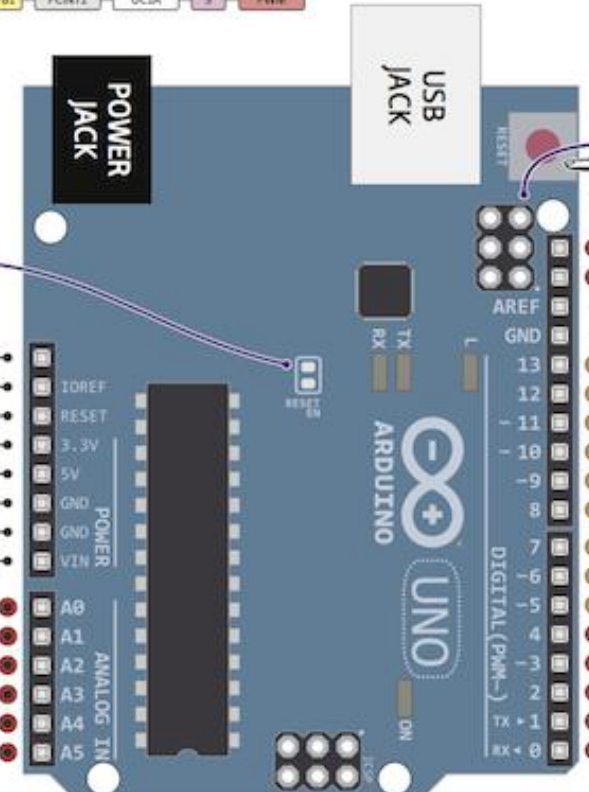


7-12V Depending on current draw

Cut to disable the auto-reset



The input voltage to the Arduino board when it is running from external power. Not USB bus power.



ver 2 rev 0 - 13.02.2013

- GND
- Power
- Control
- Physical Pin
- Port Pin
- Pin Function
- Digital Pin
- Analog Related Pin
- PWM Pin
- Serial Pin
- IDE
- Source Total 150mA

Το ARDUINO UNO

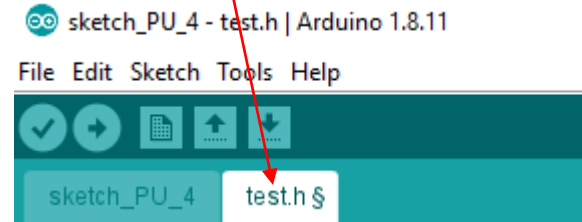
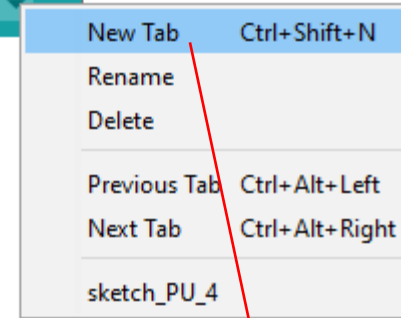
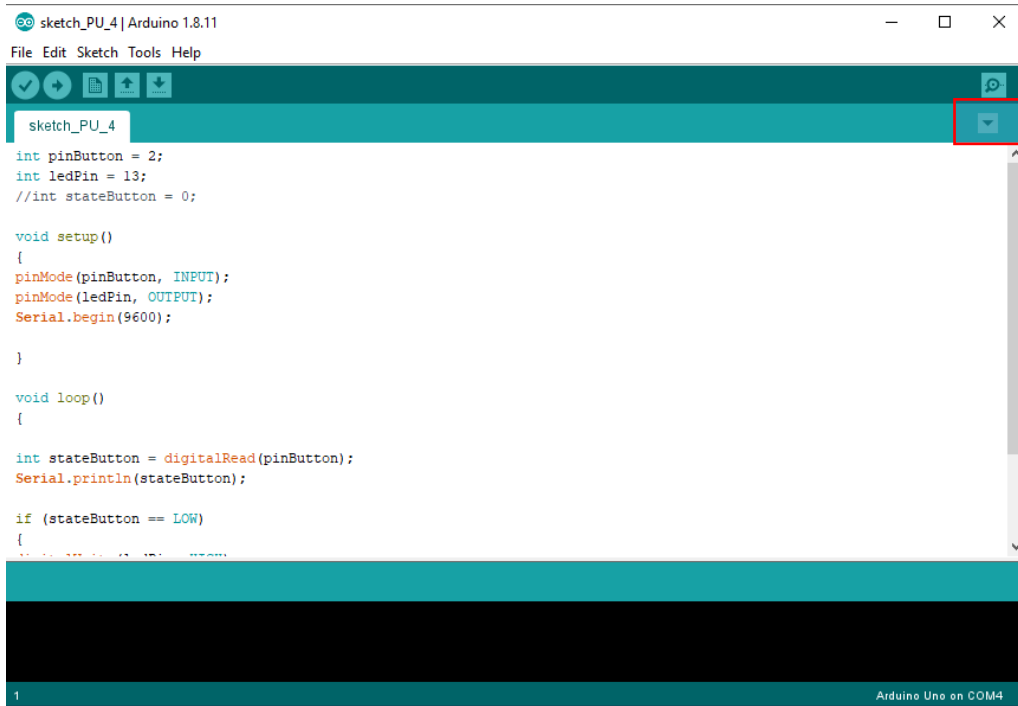
Όλες οι ακίδες αξιοποιούνται μέσω της κάρτας που φιλοξενεί τον μικροελεγκτή. Ο μικροελεγκτής μετά τον προγραμματισμό του μπορεί ακόμα και να αφαιρεθεί από την κάρτα και να τοποθετηθεί για αυτόνομη λειτουργία αφού πλαισιωθεί από τα κατάλληλα κυκλώματα.

Η κάρτα δηλαδή μπορεί να χρησιμοποιείται μόνο για τον προγραμματισμό. Φυσικά υπάρχει και η δυνατότητα τοποθέτησης ολόκληρου του Arduino στο σημείο ενδιαφέροντος για την ολοκλήρωση της εφαρμογής.

Συμπερασματικά:

- ✓ Το Arduino είναι ένα «πακέτο» που προσφέρει άμεση αξιοποίηση και προγραμματισμό ενός μικροελεγκτή.
- ✓ Οι θύρες επικοινωνίας και οι ακίδες του μικροελεγκτή προσφέρονται μέσω της κάρτας για εύκολες συνδέσεις.
- ✓ Ο προγραμματισμός γίνεται μέσω της θύρας USB.
- ✓ Η ανάπτυξη του κώδικα γίνεται σε γλώσσα τύπου C/C++.

Arduino IDE – sketch – tabs



Η βασική δομή ενός sketch

Ένα sketch αποτελείται από τρία (3) μέρη:

1. Ονοματοδοσία Μεταβλητών (μη υποχρεωτικό)

Στο πρώτο αυτό μέρος μπορούμε να δηλώσουμε μεταβλητές.

2. setup() (Υποχρεωτικό)

Το setup θα τρέξει μόνο μία φορά. Εδώ δηλώνουμε, για παράδειγμα, ποια Pin στην κάρτα είναι είσοδοι και έξοδοι.

3. loop() (Υποχρεωτικό)

Αυτό το μέρος του προγράμματος επαναλαμβάνεται συνέχεια!!!

```
// digital pin 2 has a pushbutton attached to it. Give it a name:  
int pushButton = 2;
```

```
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize serial communication at 9600 bits per second:  
  Serial.begin(9600);  
  // make the pushbutton's pin an input:  
  pinMode(pushButton, INPUT);  
}
```

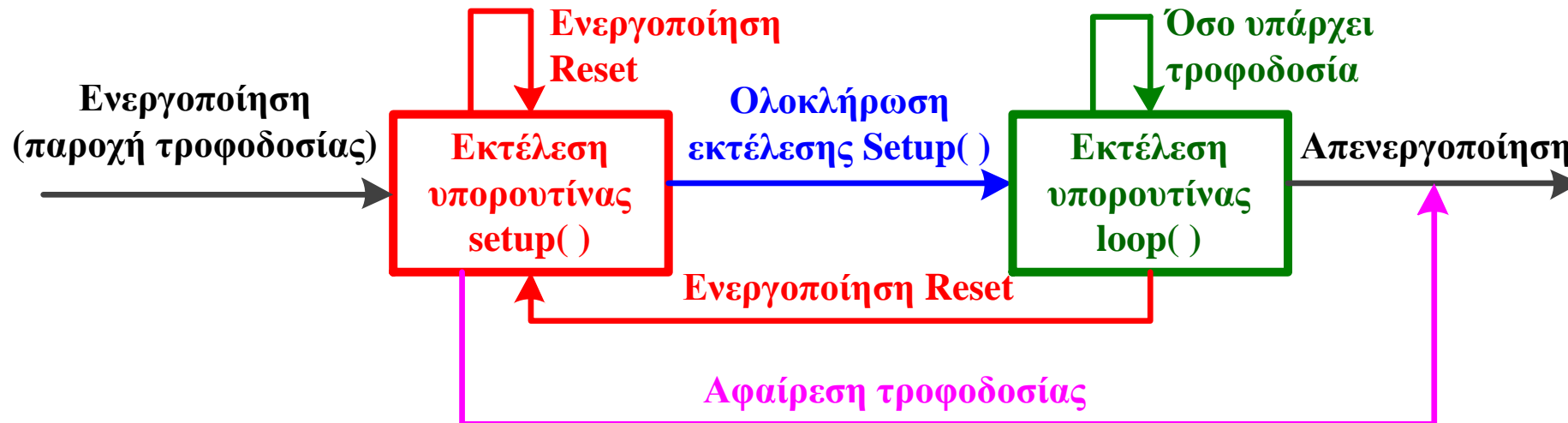
```
// the loop routine runs over and over again forever:  
void loop() {  
  // read the input pin:  
  int buttonState = digitalRead(pushButton);  
  // print out the state of the button:  
  Serial.println(buttonState);  
  delay(1);      // delay in between reads for stability  
}
```

Η βασική δομή ενός sketch

Κάθε πρόγραμμα που αναπτύσσεται σε Wiring C έχει ορισμένα βασικά συστατικά προκειμένου να είναι λειτουργικό. Έτσι, κάθε πρόγραμμα αποτελείται από δύο συναρτήσεις:

(α) τη συνάρτηση `setup()` που εκτελείται μία φορά και πάντα μετά την εφαρμογή της τροφοδοσίας ή ενεργοποίησης του Reset.

(β) τη συνάρτηση `loop()` που εκτελείται όσο το Arduino βρίσκεται σε κατάσταση λειτουργίας (παροχή τροφοδοσίας).



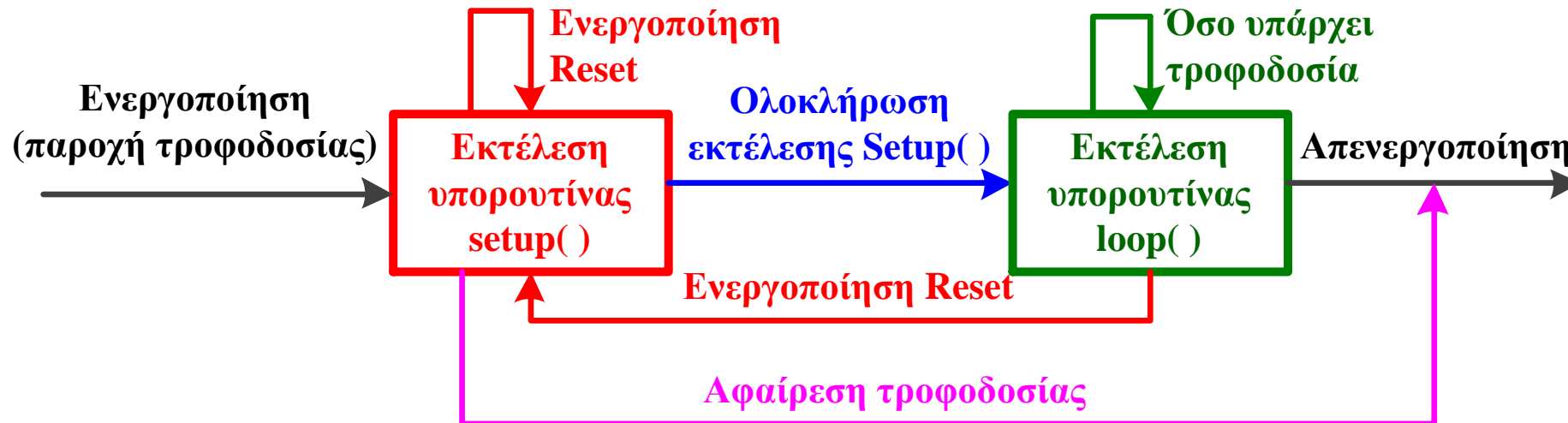
Η βασική δομή ενός sketch

Όταν ενεργοποιηθεί η τροφοδοσία ξεκινά η εκτέλεση της συνάρτησης `setup()` η οποία πραγματοποιείται μια φορά. Επαναφορά στην αρχή εκτέλεσης της `setup()` μπορεί να γίνει μόνο αν αφαιρεθεί και εφαρμοστεί ξανά η τροφοδοσία ή ενεργοποιηθεί το Reset.

Μόλις ολοκληρωθεί η εκτέλεση του κώδικα που περιλαμβάνει η `setup()`, ξεκινά η εκτέλεση του κώδικα της συνάρτησης `loop()`.

Όταν εκτελεστούν οι εντολές της `loop()`, η εκτέλεσή τους ξεκινά πάλι από την αρχή και για όσο υπάρχει τροφοδοσία στο σύστημα.

Αν ενεργοποιηθεί το Reset, γίνεται επιστροφή πάλι στην εκτέλεση του κώδικα της `setup()`.



Η βασική δομή ενός sketch

Ο κώδικας που ακολουθεί είναι ο ελάχιστος απαιτούμενος για οποιαδήποτε εφαρμογή.

//Βασική Δομή sketch

void setup()

{

//κώδικας αρχικοποίησης

}

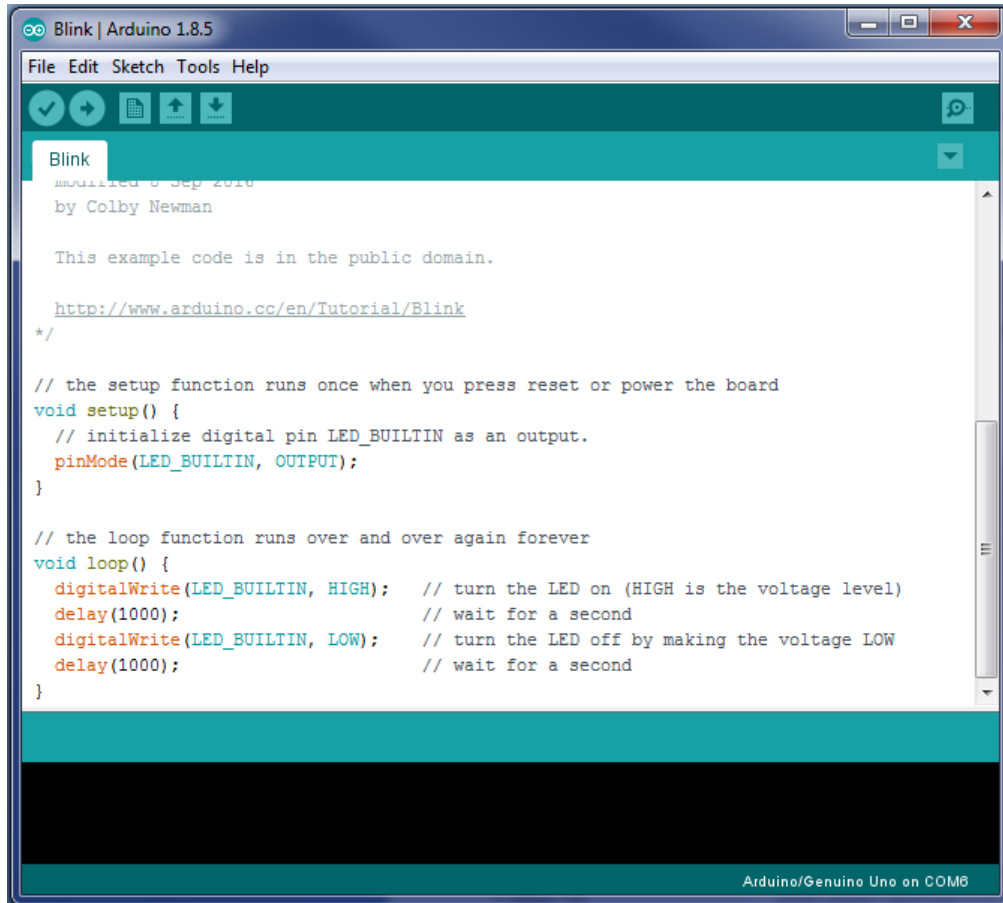
void loop()

{






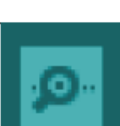
//κύριος κώδικας

}

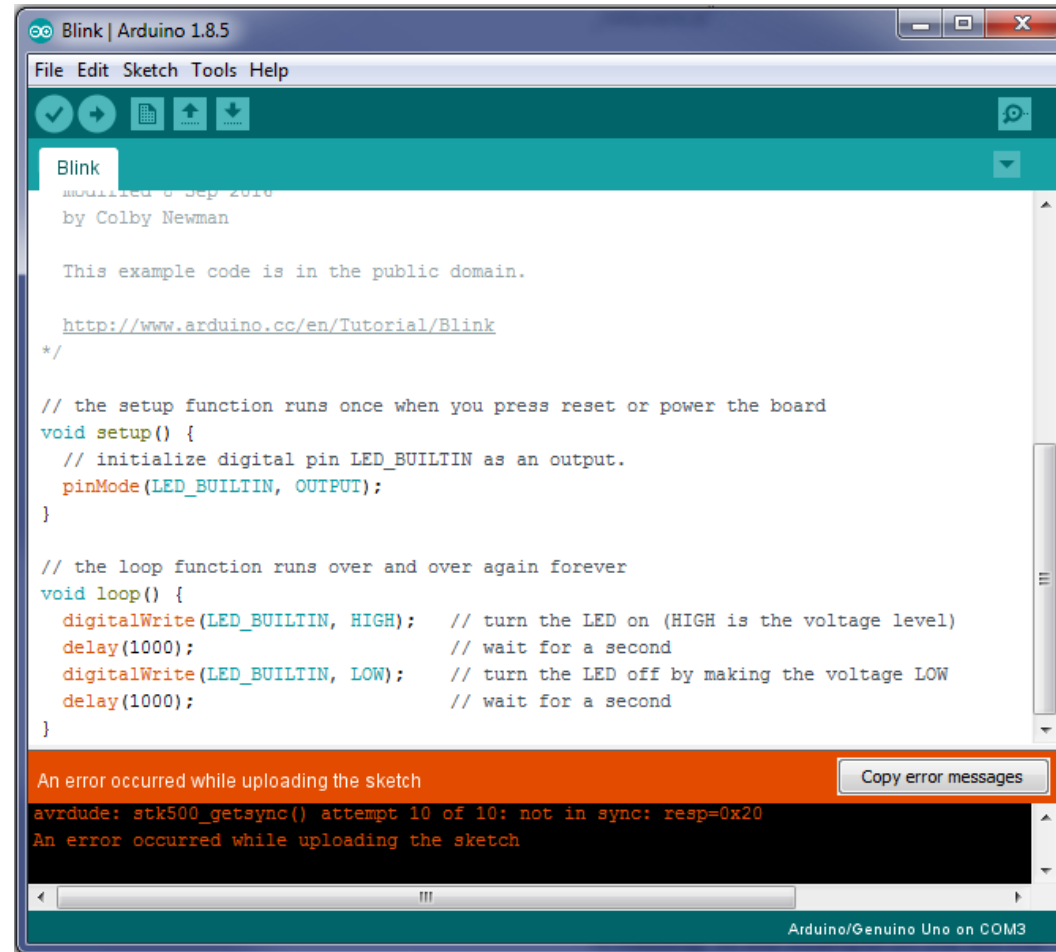
Arduino IDE



Πίνακας Β'4: Εργαλεία υπό μορφή κουμπιών στο Arduino IDE

	Verify	Ελέγχει για συντακτικά λάθη στον κώδικα.
	Upload	Μεταγλωττίζει τον κώδικα και τον φορτώνει στο Arduino.
	New	Δημιουργεί ένα νέο sketch.
	Open	Παραθέτει ένα μενού με όλα τα sketch για άνοιγμα σε νέο παράθυρο.
	Save	Αποθηκεύει ένα sketch.
	Serial Monitor	Ανοίγει την σειριακή οθόνη.

Arduino IDE – Errors

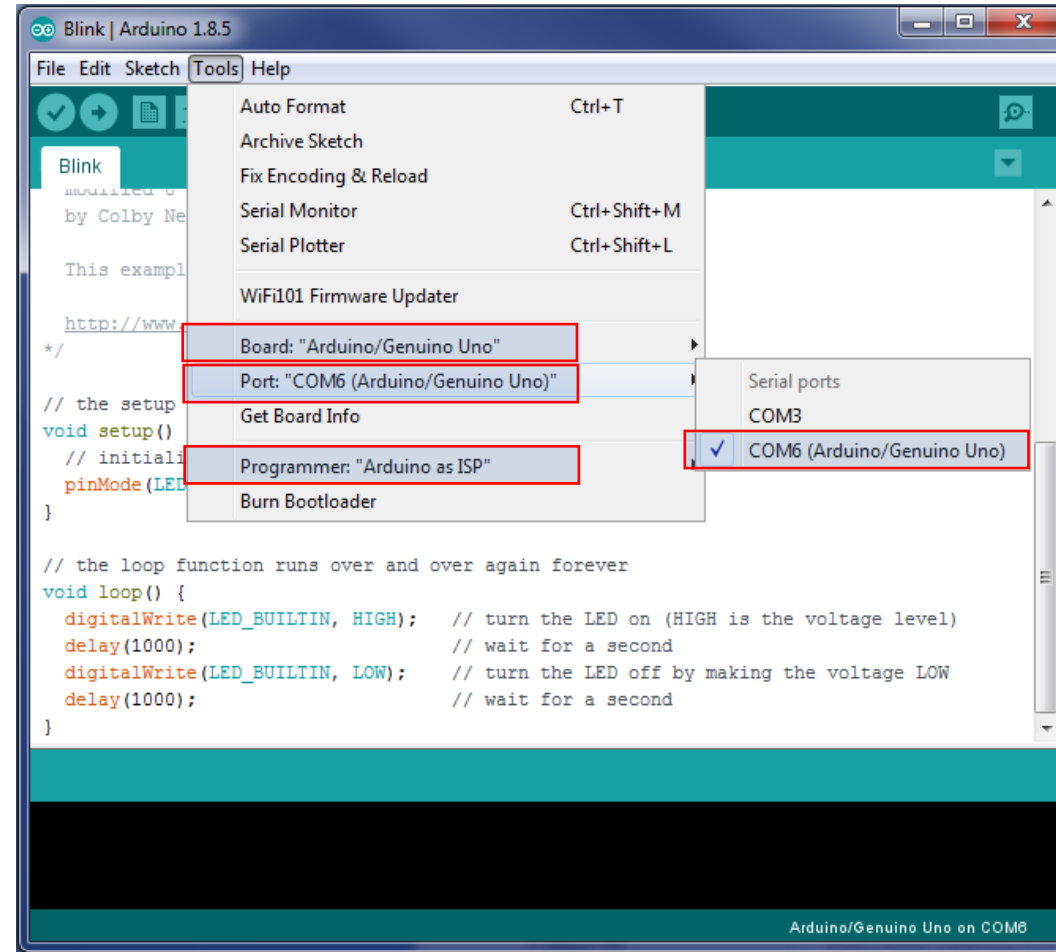


The screenshot shows the Arduino IDE window titled "Blink | Arduino 1.8.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checkmark, back, forward, upload, and download. The main editor area displays the "Blink" sketch, which includes a header comment, a setup function, and a loop function. The code is as follows:

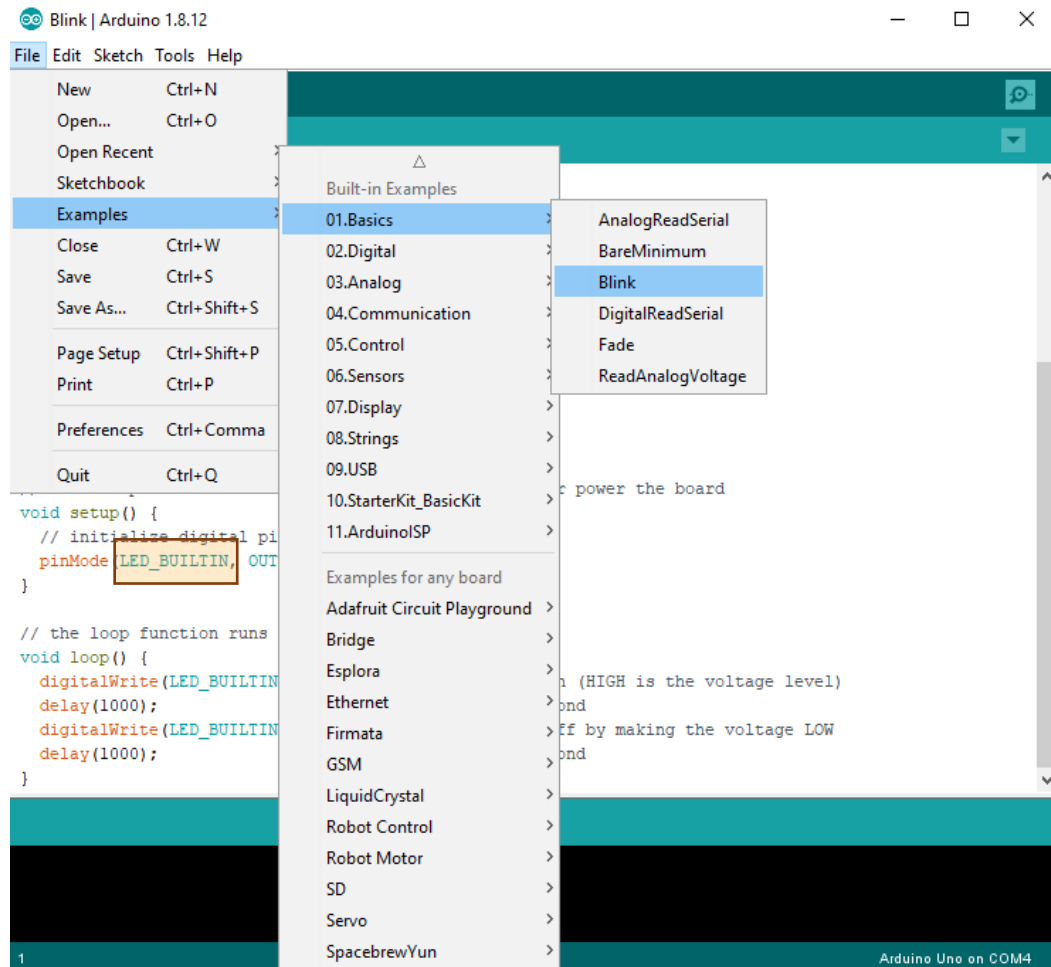
```
/*  
 * modified 8 Sep 2010  
 * by Colby Newman  
 *  
 * This example code is in the public domain.  
 *  
 * http://www.arduino.cc/en/Tutorial/Blink  
 */  
  
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

Below the code editor, an orange error message bar states: "An error occurred while uploading the sketch". To the right of this bar is a button labeled "Copy error messages". Below the error bar, the serial monitor shows the following error message: "avrduede: stk500_getsync() attempt 10 of 10: not in sync: resp=0x20". Below the serial monitor, the status bar indicates "Arduino/Genuino Uno on COM3".

Arduino IDE – Lab Setup

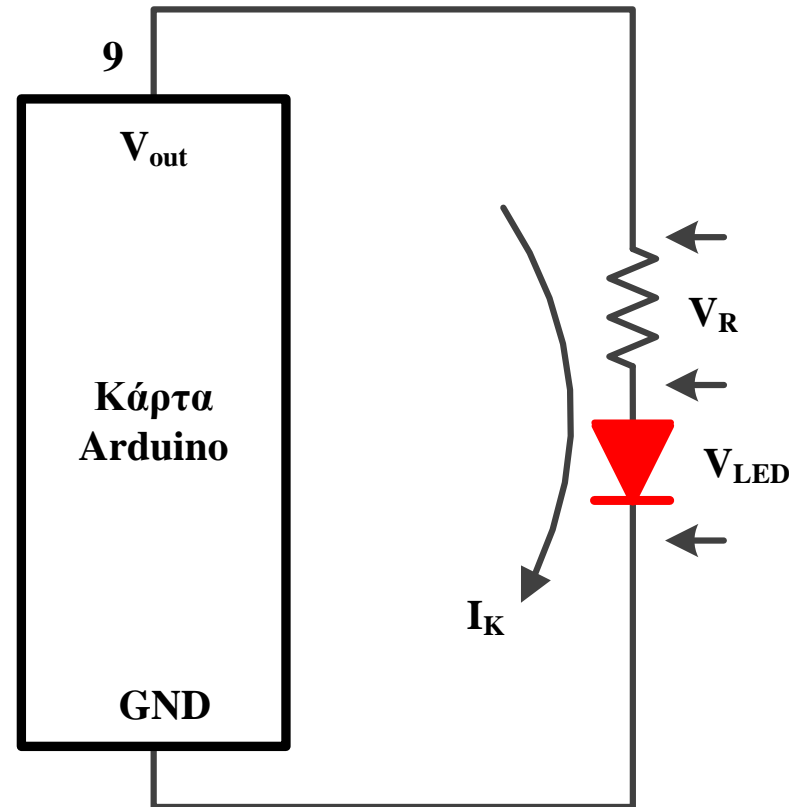


Hello Arduino! – Pin 13 (LED_BUILTIN onboard)



Hello Arduino!

Να μελετηθεί ένα κύκλωμα που θα συνδεθεί στο Arduino, θα διαθέτει ένα LED με μια αντίσταση και θα αναβοσβήνει το LED.



Hello Arduino!

Η χρήση της αντίστασης περιορισμού ρεύματος R είναι απαραίτητη προκειμένου να προφυλαχθεί η δίοδος LED από τη διαρροή υψηλού ρεύματος το οποίο θα μπορούσε να την καταστρέψει.

Το κύκλωμα θα διαρρέεται από το ρεύμα I_K .

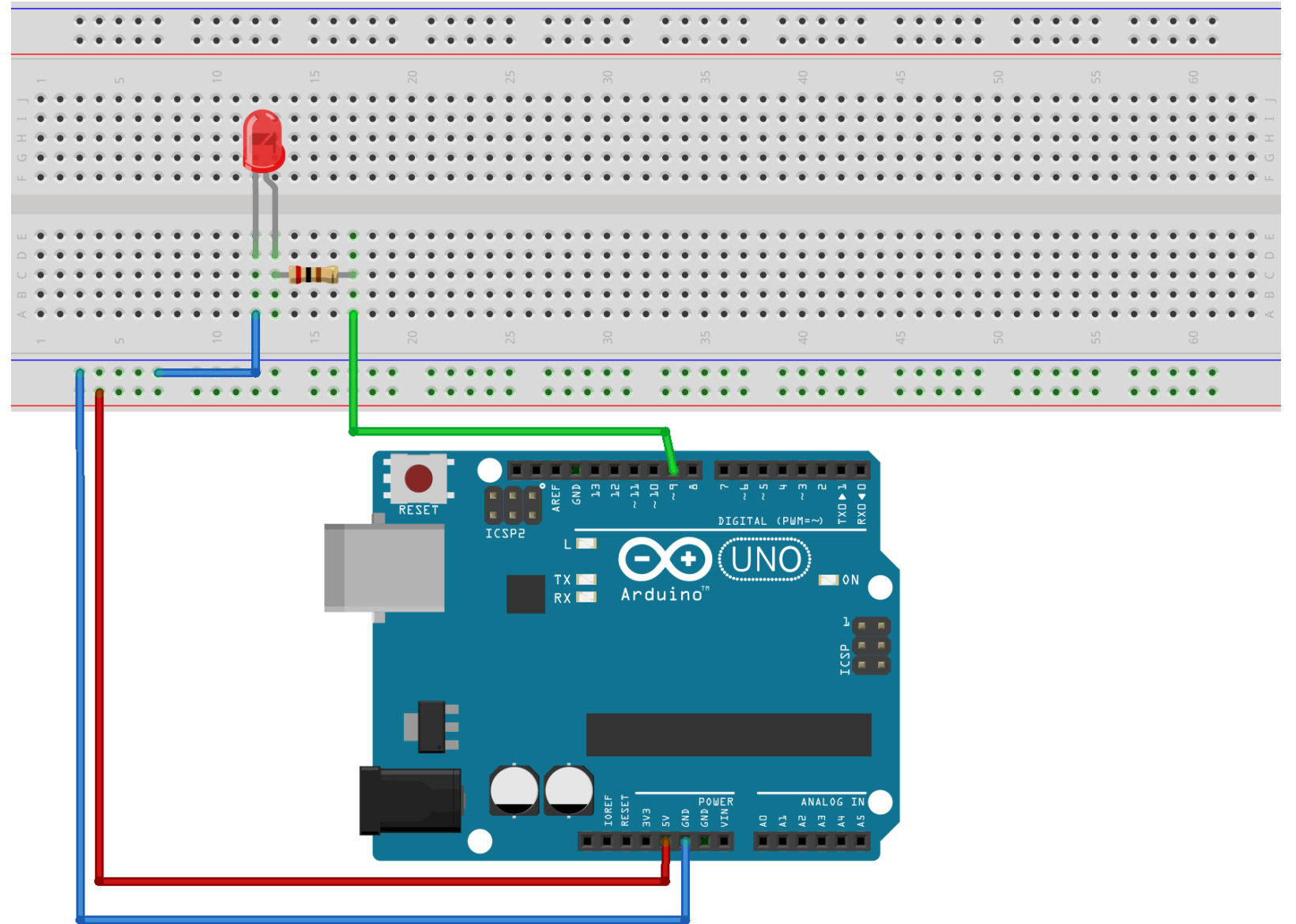
$$V_{out} = V_R + V_{LED} \quad \text{ή} \quad V_R = V_{out} - V_{LED}$$

Επομένως:

$$I_K = \frac{V_R}{R} = \frac{V_{out} - V_{LED}}{R}$$

$$I_K = \frac{5V - 2.1V}{200\Omega} = \frac{2.9V}{200\Omega} = 0.0145A = 14.5mA$$

Hello Arduino!



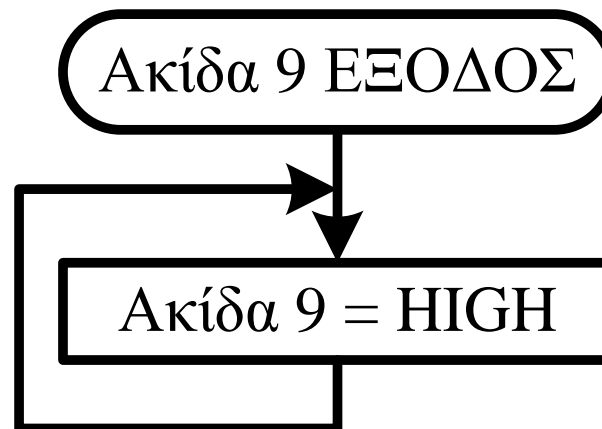
fritzing

Hello Arduino! – Λογικό Διάγραμμα

Ο έλεγχος του ρεύματος στο κύκλωμα γίνεται μέσω κατάλληλων εντολών και προϋποθέτει δύο ενέργειες:

(α) καθορισμό ψηφιακού ακροδέκτη ως εξόδου.

(β) καθορισμό στάθμης σήματος (0 ή 5V) στον επιλεγμένο ακροδέκτη.



Διαγράμματα Ροής – Flowcharts

Το διάγραμμα ροής είναι μια διαγραμματική αναπαράσταση ενός αλγορίθμου.

Το διάγραμμα ροής είναι πολύ χρήσιμο στο να γράφετε προγράμματα και να εξηγείτε αυτά σε άλλους.

Σύμβολα που χρησιμοποιούνται στο διάγραμμα ροής

Διαφορετικά σύμβολα χρησιμοποιούνται για διαφορετικές καταστάσεις στα διαγράμματα ροής, για παράδειγμα:

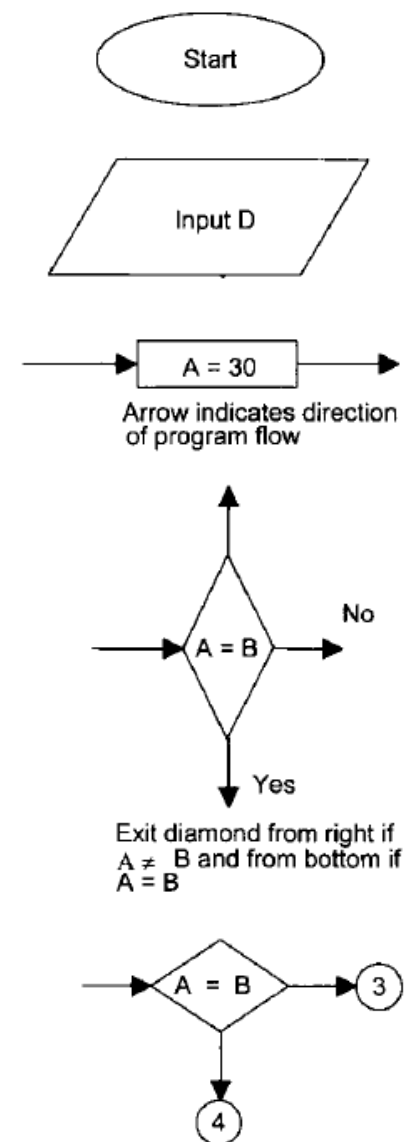
Η είσοδος / έξοδος και η λήψη αποφάσεων έχουν διαφορετικά σύμβολα.

Ο παρακάτω πίνακας περιγράφει όλα τα σύμβολα που χρησιμοποιούνται για τη δημιουργία ροής.

Διαγράμματα Ροής – Flowcharts

Σύμβολο	Σκοπός		Περιγραφή
	Flow line	Γραμμή ροής/βέλος	Χρησιμοποιείται για να υποδεικνύει τη ροή της λογικής συνδέοντας τα σύμβολα.
	Terminal (Start/Stop)	Τερματικό (Start / Stop)	Χρησιμοποιείται για να αντιπροσωπεύει την αρχή και το τέλος του διαγράμματος ροής.
	Input / Output	Είσοδος / Έξοδος	Χρησιμοποιείται για λειτουργία εισόδου και εξόδου.
	Processing	Στάδια Επεξεργασίας	Χρησιμοποιείται για αριθμητικές πράξεις και χειρισμούς δεδομένων.
	Decision	Υποθέσεις/Αποφάσεις	Χρησιμοποιείται για να αντιπροσωπεύει τη λειτουργία στην οποία υπάρχουν δύο εναλλακτικές λύσεις, αληθείς και ψευδείς.
	On-page Connector	Σύνδεσμος επί της σελίδας	Χρησιμοποιείται για να ενταχθεί σε διαφορετική γραμμή ροής
	Off-page Connector	Συνδετήρας εκτός σελίδας	Χρησιμοποιείται για τη σύνδεση τμήματος ροής σε διαφορετική σελίδα.
	Predefined Process/Function	Προκαθορισμένη διαδικασία / λειτουργία	Χρησιμοποιείται για να αντιπροσωπεύει μια ομάδα δηλώσεων που εκτελούν μια εργασία επεξεργασίας.

Παραδείγματα



Hello Arduino!

Η εντολή που ρυθμίζει τη μορφή του ακροδέκτη είναι `pinMode` που συντάσσεται ως:

```
pinMode(pin, mode);
```

όπου **pin** είναι ο ψηφιακός ακροδέκτης (οι διαθέσιμοι ψηφιακοί ακροδέκτες μπορεί να διαφέρουν ανάλογα με την έκδοση του Arduino) και

mode ο τρόπος που θα χρησιμοποιηθεί (οι βασικότερες επιλογές είναι, **INPUT** για είσοδο και **OUTPUT** για έξοδο).

Στο συγκεκριμένο παράδειγμα, επιλέχθηκε ο ψηφιακός ακροδέκτης 9 και ο καθορισμός του ως εξόδου θα γίνει ως εξής:

```
pinMode(9,OUTPUT);
```

Hello Arduino!

Η εντολή που ρυθμίζει το αν θα ανάβει ή θα είναι σβηστό το LED συντάσσεται ως:

digitalWrite(pin, status);

όπου **pin** είναι ο ψηφιακός ακροδέκτης (οι διαθέσιμοι ψηφιακοί ακροδέκτες μπορεί να διαφέρουν ανάλογα με την έκδοση του Arduino) και

status είναι η στάθμη του σήματος στην ακίδα (0V για LOW και 5V για HIGH).

Επομένως η εντολή που θα ανάψει το LED θα είναι η:

digitalWrite(9, HIGH);

Και η εντολή που θα σβήσει το LED θα είναι η:

digitalWrite(9, LOW);

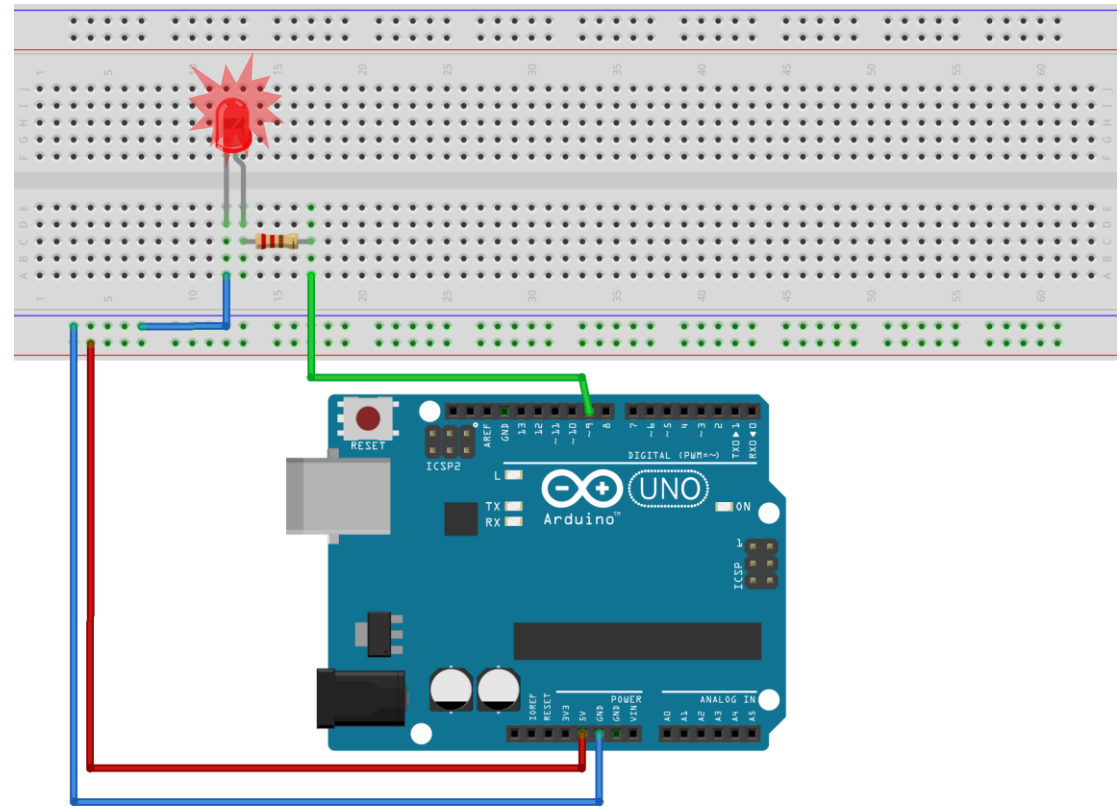
Hello Arduino! – Wiring C

```
void setup()  
{  
  pinMode(9 , OUTPUT);  
}
```

```
void loop()  
{  
  digitalWrite(9 , HIGH);  
}
```

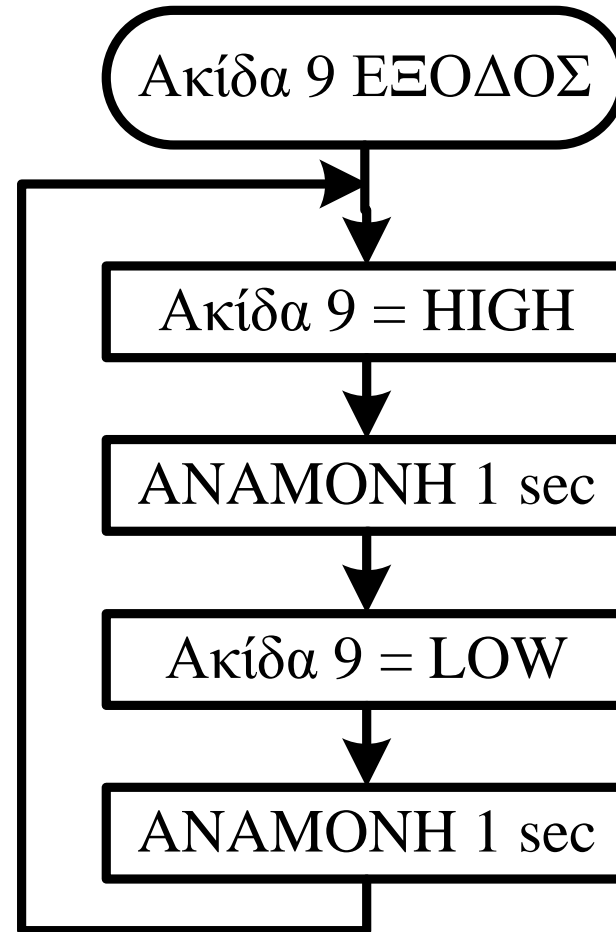
Blinking Led – 1

Με βάση την προηγούμενη άσκηση να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα αναβοσβήνει ένα LED.



fritzing

Blinking Led – 1 – Λογικό Διάγραμμα



Blinking Led – 1 – Wiring C

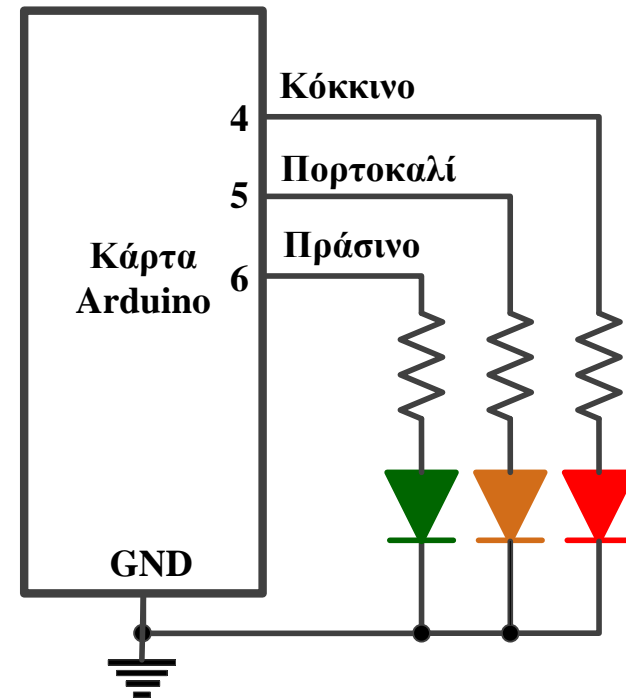
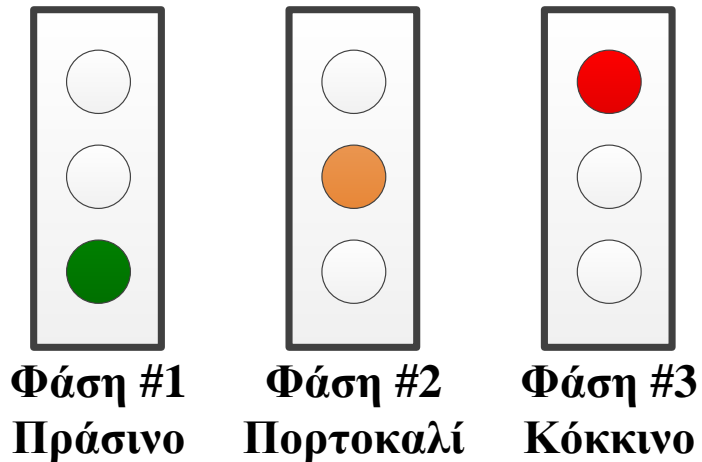
```
void setup()  
{  
  pinMode(9 , OUTPUT);  
}
```

```
void loop()  
{  
  digitalWrite(9 , HIGH);  
  delay(1000);  
  digitalWrite(9 , LOW);  
  delay(1000);  
}
```

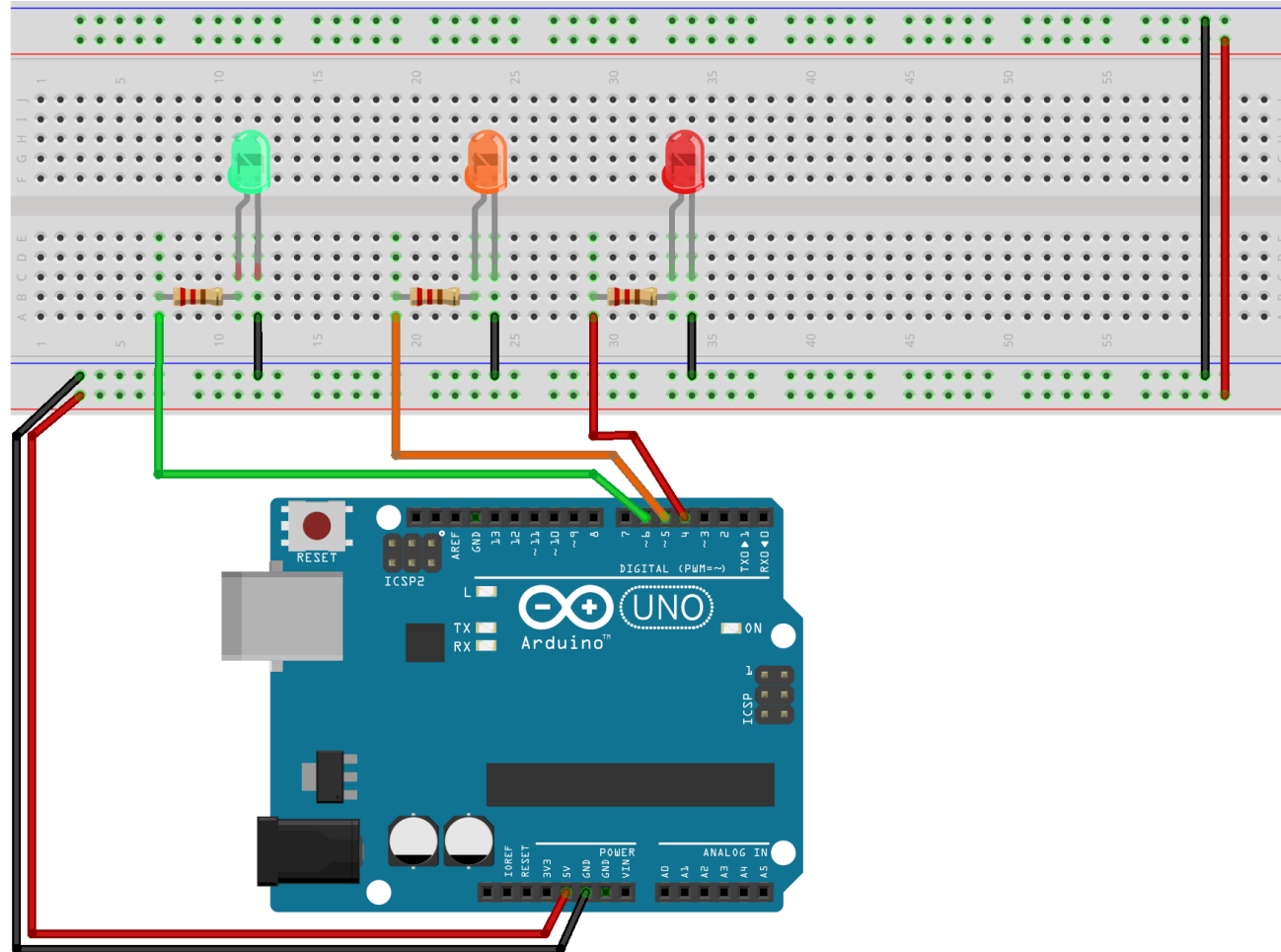

Blinking Led – 2

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα ανάβει τα LED ενός φωτεινού σηματοδότη με βάση τους παρακάτω χρόνους:

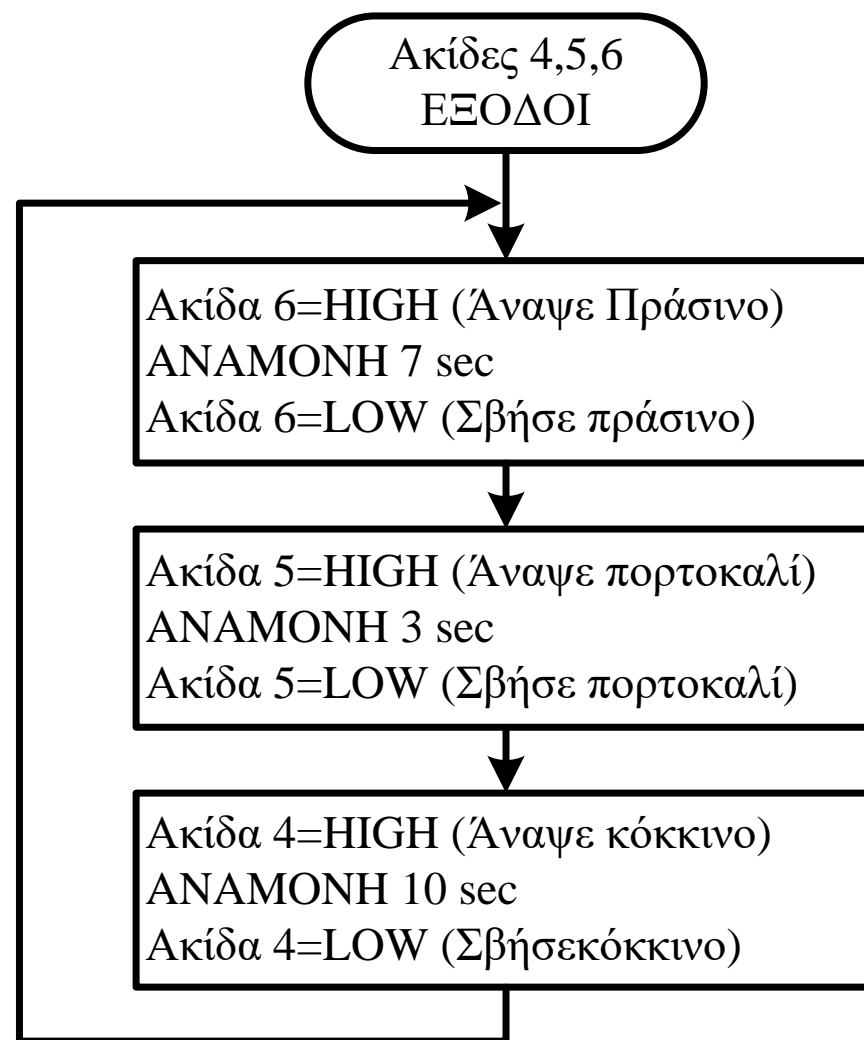
Πράσινο 7 sec, Πορτοκαλί 3 sec, Κόκκινο 10 sec



Blinking Led – 2



Blinking Led – 2 – Λογικό Διάγραμμα



Blinking Led – 2 – Wiring C

```
int RledPin = 4;           // Ονόμασε ακίδα 4 RledPin
int OledPin = 5;          // Ονόμασε ακίδα 5 OledPin
int GledPin = 6;          // Ονόμασε ακίδα 6 GledPin

unsigned int RDelay = 10000; //Χρονική καθυστέρηση για το κόκκινο
unsigned int ODelay= 3000;   //Χρονική καθυστέρηση για το πορτοκαλί
unsigned int GDelay = 7000;  //Χρονική καθυστέρηση για το πράσινο

void setup()              //Συνάρτηση αρχικοποίησης
{
  pinMode(RledPin, OUTPUT); //Κάνε τις ακίδες
  pinMode(OledPin, OUTPUT); //4, 5 και 6
  pinMode(GledPin, OUTPUT); //εξόδους
}
```

Blinking Led – 2 – Wiring C

```
void loop()
{
  digitalWrite(GledPin, HIGH);
  delay(GDelay);
  digitalWrite(GledPin, LOW);

  digitalWrite(OledPin, HIGH);
  delay(ODelay);
  digitalWrite(OledPin, LOW);

  digitalWrite(RledPin, HIGH);
  delay(RDelay);
  digitalWrite( RledPin, LOW);
}

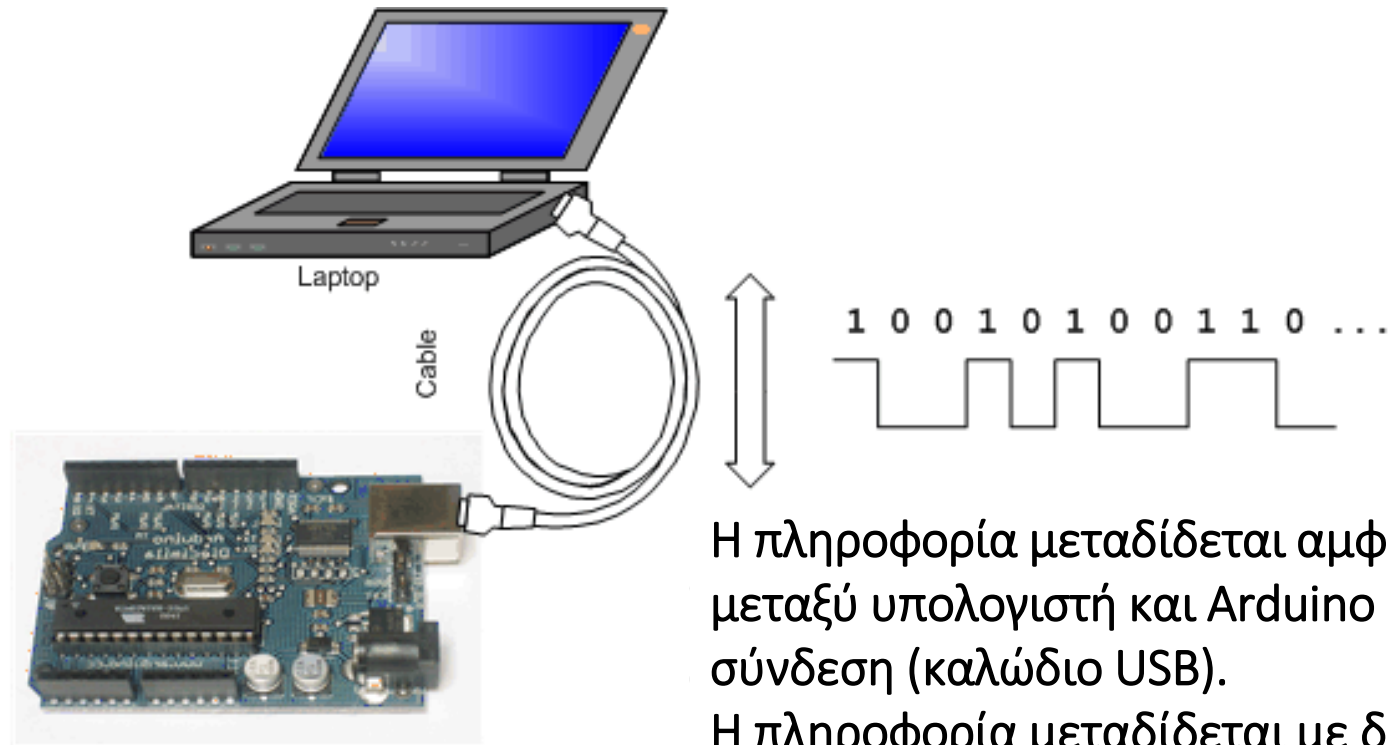
//Κύριος βρόχος

//Άναψε το πράσινο LED
//για 7sec και
//σβήσε το.

//Άναψε το πορτοκαλί LED
//για 3sec και
//σβήσε το.

//Άναψε το πράσινο LED
//για 10sec και
//σβήσε το.
```

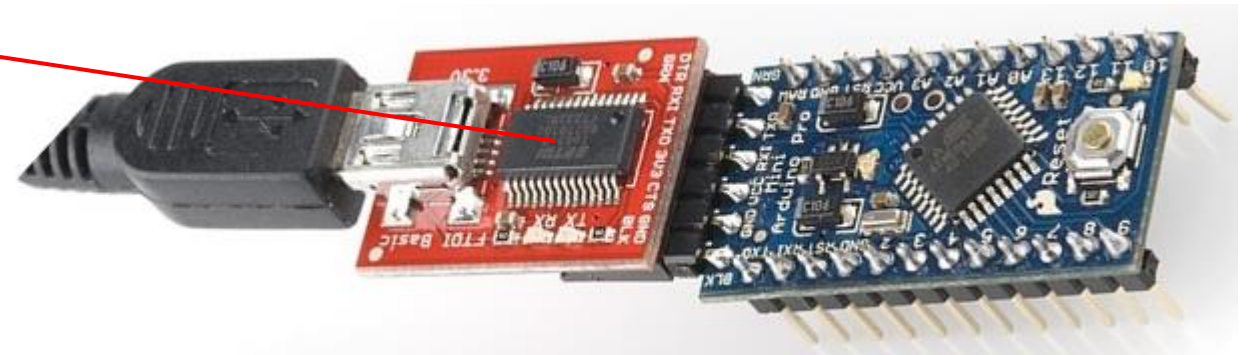
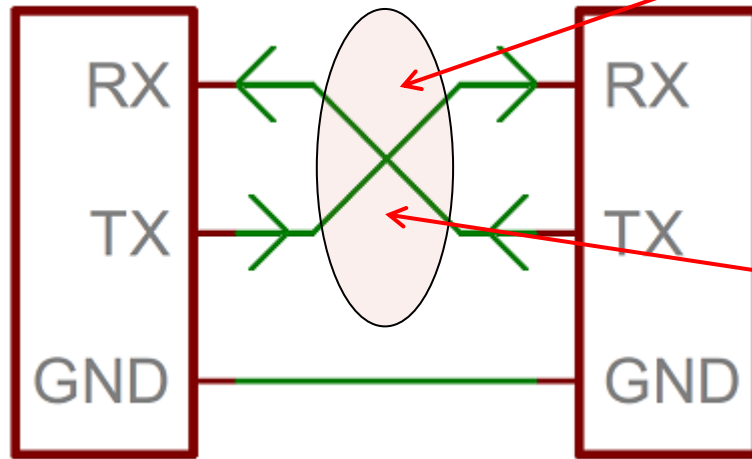
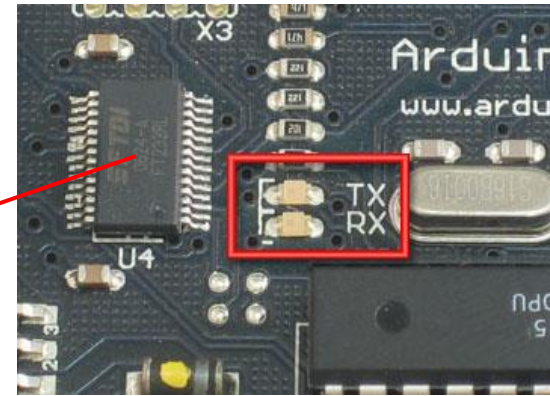
Σειριακή Επικοινωνία – Serial Com



Η πληροφορία μεταδίδεται αμφίδρομα μεταξύ υπολογιστή και Arduino με σειριακή σύνδεση (καλώδιο USB).

Η πληροφορία μεταδίδεται με δυαδικά ψηφία.

Σειριακή Επικοινωνία – Serial Com



Η ταχύτητα της σειριακής επικοινωνίας μετράτε σε **Baud rate**.

Ένα (1) baud σημαίνει μετάδοση ενός (1) συμβόλου κάθε δευτερόλεπτο.

Ένα σύμβολο μπορεί να είναι περισσότερο από 1 bit.

Συνήθως μεταδίδεται ένας χαρακτήρας **ASCII (8 bit)**.

Γνωστά baud rate είναι: 1200, 2400, 4800, 9600, 19200, 38400, 57600 και 115200.

Σειριακή Επικοινωνία

Ας υποθέσουμε ότι θέλουμε να αποστείλουμε με 9600 baud rate, 8 data bits, με 1 start και 1 stop bit. Μια συσκευή μεταδίδει τους ASCII χαρακτήρες 'Ο' και 'Κ' και θα πρέπει να δημιουργήσει δύο πακέτα δεδομένων. Η δεκαδική τιμή του ASCII 'Ο' μια και είναι κεφαλαίο είναι το 79, το οποίο στο δυαδικό είναι 01001111, ενώ το 'Κ' είναι το 75 επομένως 01001011. Φυσικά πρέπει να λάβουμε και τα bit συγχρονισμού δηλαδή το start και το stop bit, δηλαδή στο σύνολο 10 ψηφία. Τέλος, θεωρούμε ότι πρώτα μεταδίδονται τα λιγότερα σημαντικά δυαδικά ψηφία.



Αφού μεταδίδουμε σε 9600 bps (bit per second), ο χρόνος που χρειάζεται κάθε bit είναι $1/(9600 \text{ bps})$ ή 104 μs για κάθε bit. Για κάθε byte δεδομένων που μεταδίδεται, στην πραγματικότητα μεταδίδουμε 10 bits: το start bit, τα 8 data bits, και το stop bit. Επομένως στα 9600 bps, στέλνουμε 9600 bits κάθε δευτερόλεπτο ή 960 (9600/10) bytes το δευτερόλεπτο.

Arduino ASCII Table:

<https://www.arduino.cc/en/Tutorial/ASCIITable>

Βιβλιοθήκη – Serial()



Functions

```
If (Serial)
available()
availableForWrite()
begin()
end()
find()
findUntil()
flush()
parseFloat()
parseInt()
peek()
print()
println()
read()
readBytes()
readBytesUntil()
setTimeout()
write()
serialEvent()
```

Η βιβλιοθήκη **Serial** στο Arduino είναι μια συλλογή από χρήσιμες συναρτήσεις που μπορούμε να καλέσουμε μέσα σε ένα sketch και να χρησιμοποιήσουμε το **Serial Monitor** για να επιτύχουμε επικοινωνία και ανταλλαγή πληροφοριών μεταξύ του υπολογιστή και του Arduino.

Serial()



- Η σειριακή επικοινωνία χρησιμοποιείται για την επικοινωνία μεταξύ του Arduino board και του υπολογιστή ή άλλων συσκευών.
- Όλα τα Arduino boards έχουν τουλάχιστον μία σειριακή πόρτα Serial Port (UART ή USART).
- Επικοινωνεί και με τους ψηφιακούς ακροδέκτες 0 (RX) και 1 (TX) όπως και με τον υπολογιστή μέσω του USB.
- Με τη σύνδεση USB γίνεται και η αποθήκευση του εκτελέσιμου αρχείου του προγράμματος στη μνήμη του μικροελεγκτή.
- Αν χρησιμοποιείς τη σύνδεση USB και το ενσωματωμένο Serial Monitor δε μπορείς να χρησιμοποιήσεις τους ακροδέκτες 0 και 1 για ψηφιακή είσοδο και έξοδο.
- Σε περίπτωση που χρησιμοποιείς τη βιβλιοθήκη Serial, άνοιξε το Serial Monitor και επέλεξε το ίδιο baud rate που δήλωσες και στην κλήση του begin() στο Setup().
- Στα Arduino boards, η σειριακή στους ακροδέκτες TX/RX χρησιμοποιεί TTL logic levels (5V or 3.3V ανάλογα με την έκδοση). Μη συνδέετε αυτά τους ακροδέκτες απευθείας σε μια RS232 serial port γιατί λειτουργεί σε +/- 12V και μπορεί να καταστρέψει το Arduino board.

Serial Monitor – IDE

Open Serial Monitor

The image shows the Arduino IDE interface with the Serial Monitor window open. The Tools menu is open, highlighting the Serial Monitor option. The Serial Monitor window is titled "COM6 (Arduino/Genuino Uno)" and contains a text input field, a "Send" button, and a "Clear" button. The Serial Monitor window also has a "Line Ending" dropdown menu set to "No line ending" and a "9600 baud" dropdown menu. The IDE window shows the code for a sketch named "sketch_dec15b" with the following code:

```
const int ledPin = 13;
unsigned int ledOnTime = 1000;
unsigned int ledOffTime = 1000;

void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
  delay(50); //give time to initialize serial
  // initialize the ledPin as an output:
  pinMode(ledPin, OUTPUT);
  Serial.println("Set monitor to No line ending");
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(ledOnTime);
  digitalWrite(ledPin, LOW);
  delay(ledOffTime);
}
```

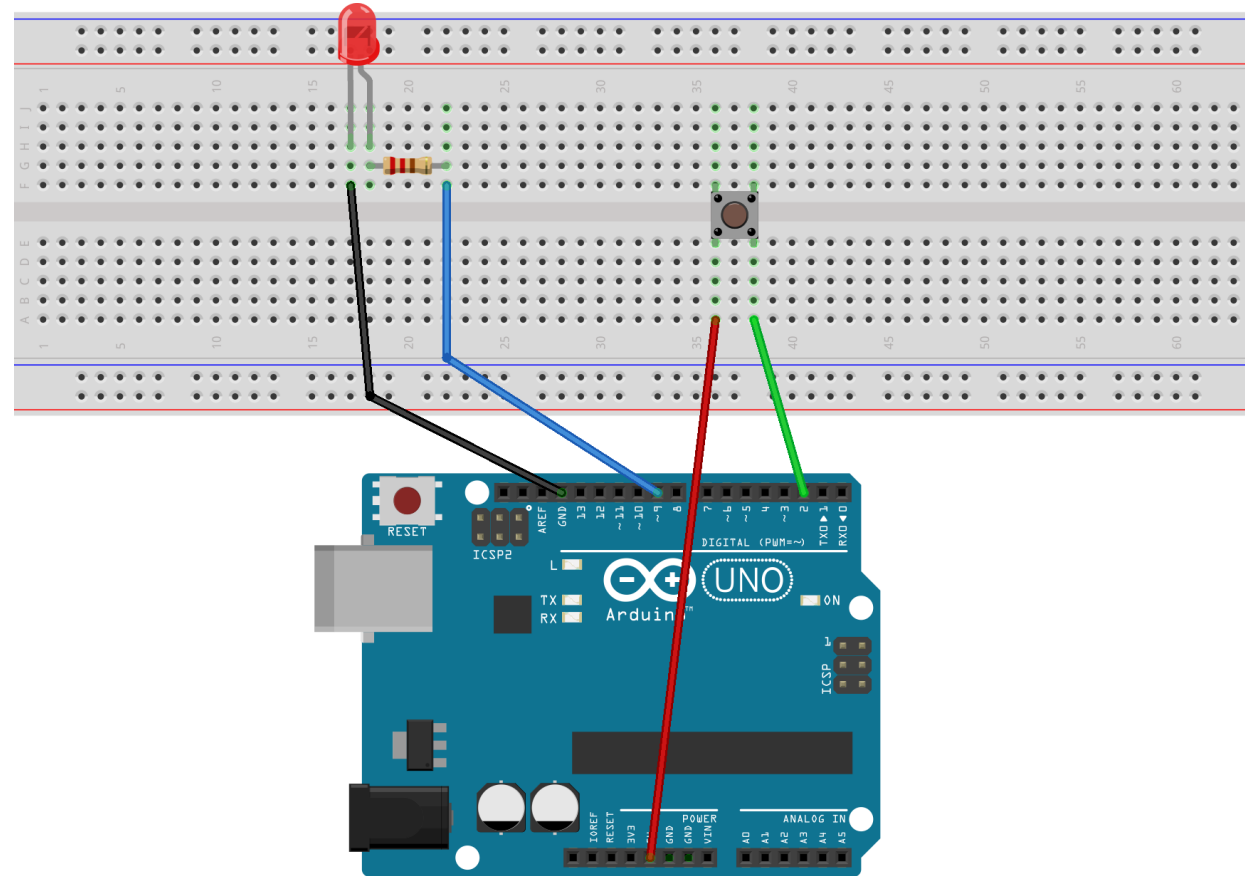
The IDE window also shows the status bar at the bottom indicating "Arduino/Genuino Uno on COM6".

Annotations in the image include:

- Open Serial Monitor**: A yellow box with an arrow pointing to the Serial Monitor option in the Tools menu.
- Input data**: A yellow box with an arrow pointing to the text input field in the Serial Monitor window.
- Send data**: A yellow box with an arrow pointing to the "Send" button in the Serial Monitor window.
- Serial Monitor**: A yellow box with an arrow pointing to the Serial Monitor window title bar.
- Clear**: A yellow box with an arrow pointing to the "Clear" button in the Serial Monitor window.
- Uncheck to stop scroll**: A yellow box with an arrow pointing to the "Autoscroll" checkbox in the Serial Monitor window.
- Line Ending**: A yellow box with an arrow pointing to the "Line Ending" dropdown menu in the Serial Monitor window.
- Baud Rate**: A yellow box with an arrow pointing to the "9600 baud" dropdown menu in the Serial Monitor window.

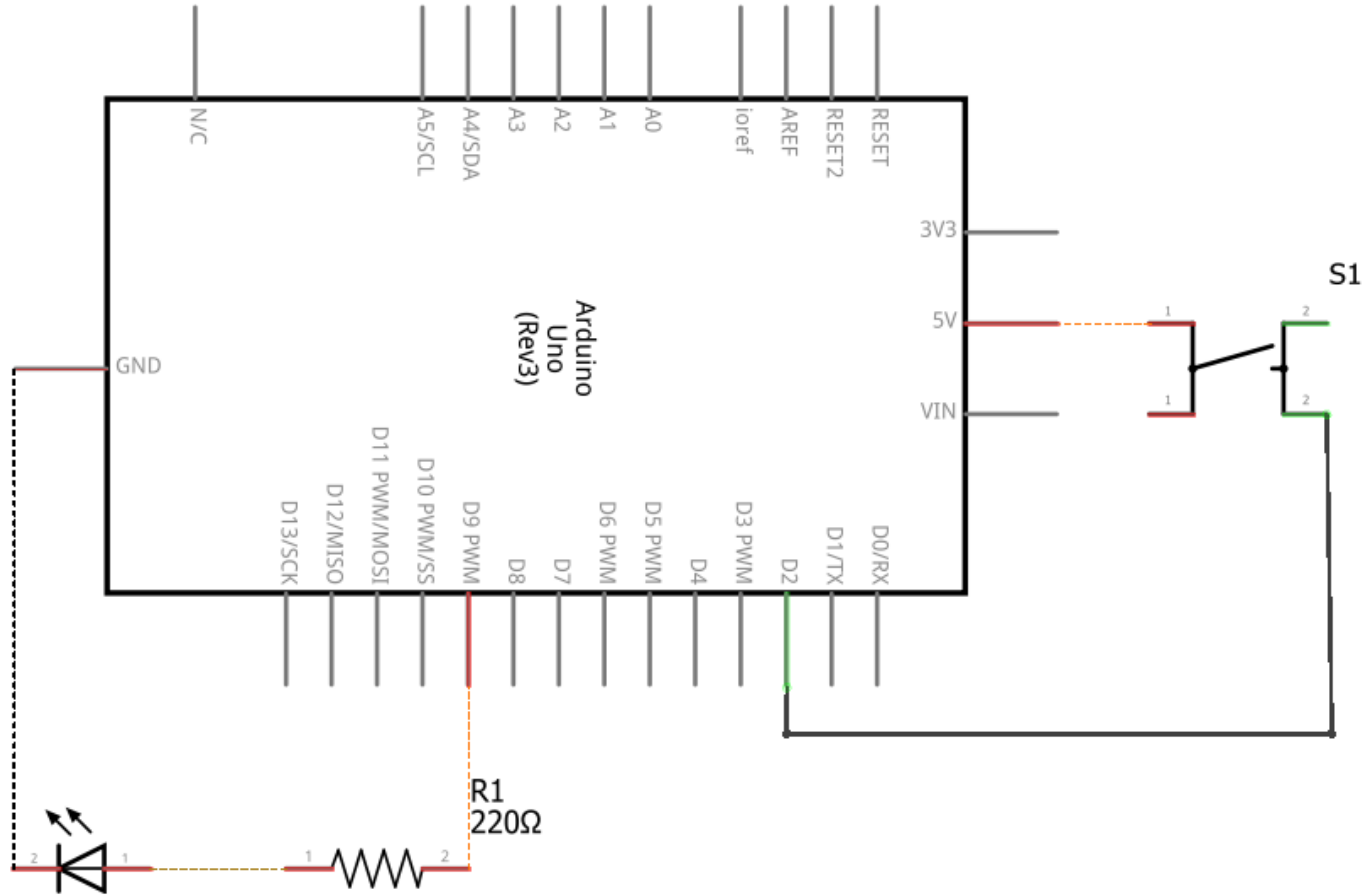
Pushing Led – 1 – Floating

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα εμφανίζει στη σειριακή την κατάσταση ενός διακόπτη ώθησης (push button) που ελέγχει ένα LED.



fritzing

Pushing Led – 1 – Floating



Pushing Led – 1 – Floating

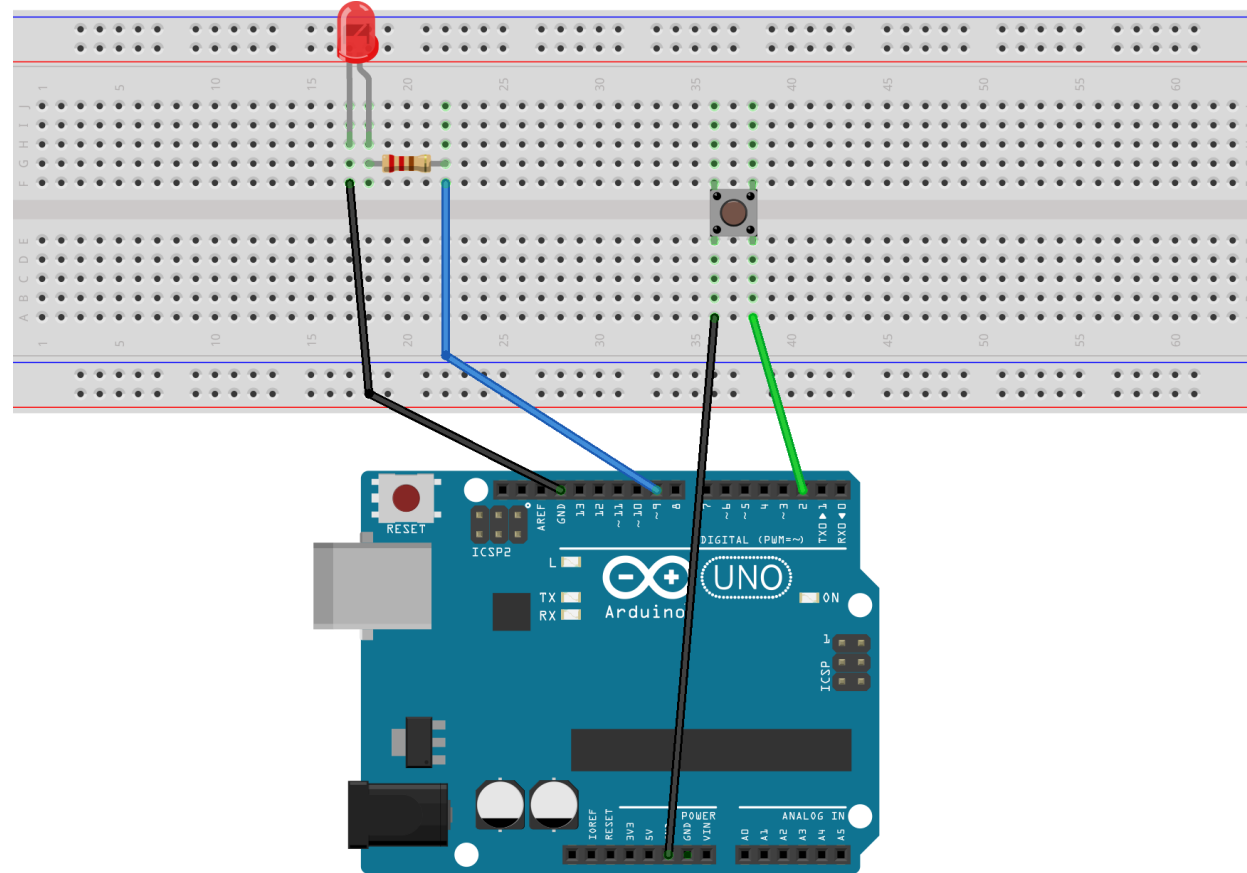
```
int pinButton = 2;           //Ονόμασε ακίδα 2 pinButton
int ledPin = 9;             //Ονόμασε ακίδα 9 ledPin

void setup()                //Συνάρτηση αρχικοποίησης
{
  pinMode( pinButton, INPUT ); //Κάνε ακίδα 2 είσοδο
  pinMode( ledPin, OUTPUT );   //και ακίδα 9 έξοδο
  Serial.begin(9600);         //ενεργοποιούμε σειριακή με ταχύτητα 9600bps
}

void loop(){
  int stateButton = digitalRead(pinButton); //εντολή ανάγνωσης τιμής από ψηφιακή είσοδο
  Serial.println(stateButton);
  if (stateButton == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
    digitalWrite(ledPin, LOW);
}
```


Pushing Led – 2 – Pull-up Resistors

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα εμφανίζει στη σειριακή την κατάσταση ενός διακόπτη ώθησης (push button) με ενεργοποιημένες τις Pull-up resistors στον διακόπτη.



fritzing

Pushing Led – 2 – Pull-up Resistors



Pushing Led – 2 – Pull-up Resistors

```
int pinButton = 2;           //Ονόμασε ακίδα 2 pinButton
int ledPin = 9;             //Ονόμασε ακίδα 9 ledPin

void setup()                //Συνάρτηση αρχικοποίησης
{
  pinMode(pinButton, INPUT_PULLUP); //Κάνε ακίδα 2 είσοδο με Pull-up
  pinMode(ledPin, OUTPUT);      //και ακίδα 9 έξοδο
  Serial.begin(9600);          //ενεργοποιούμε σειριακή με ταχύτητα 9600bps
}

void loop(){
  int stateButton = digitalRead(pinButton);
  Serial.println(stateButton);
  if (stateButton == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
    digitalWrite(ledPin, LOW);
}
```

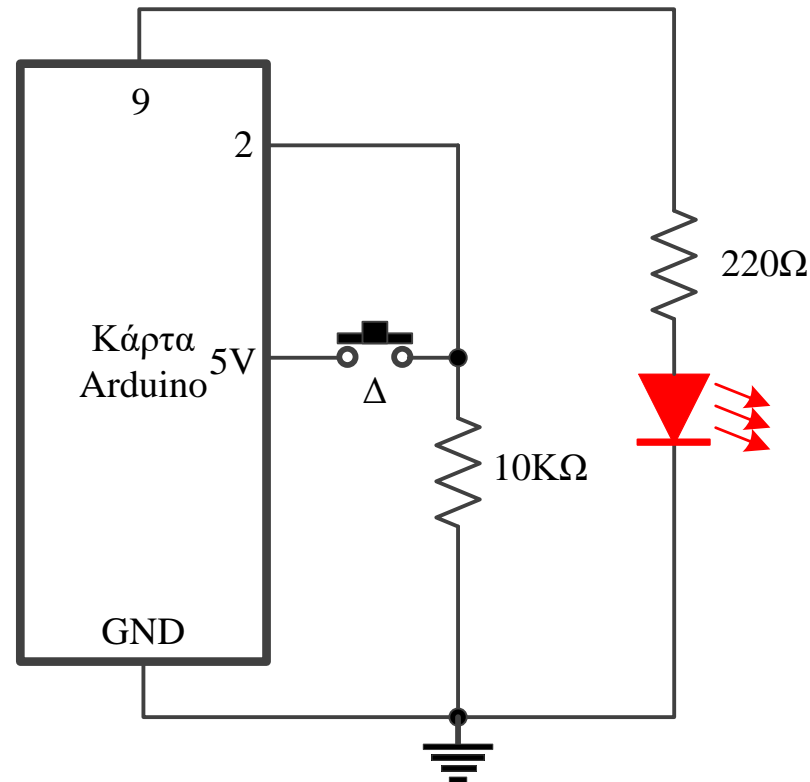
Done uploading.

Sketch uses 900 bytes (2%) of program storage space. Maximum is 32256 bytes.

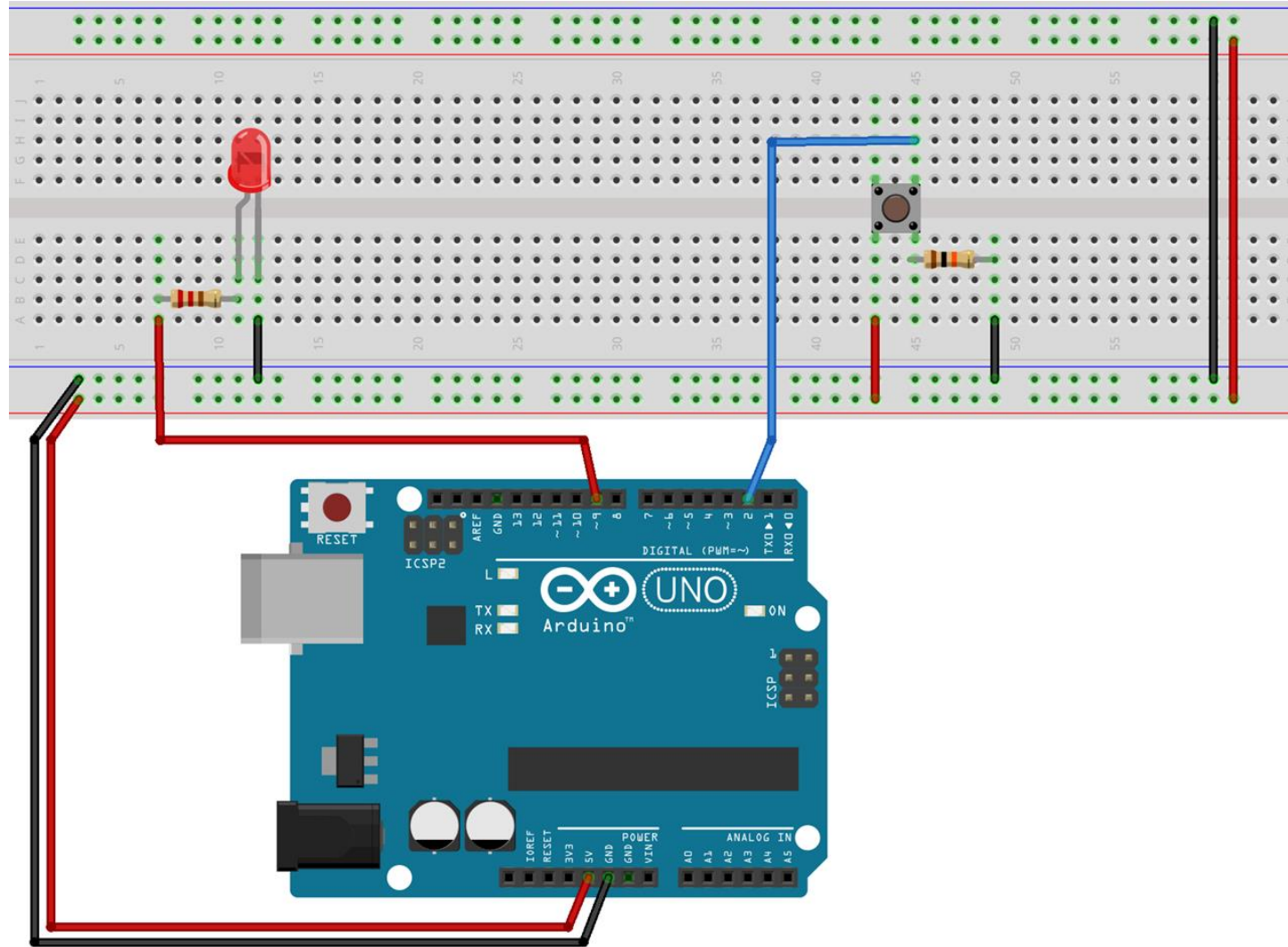
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

Pushing Led – 3 – External Pull-down Resistor

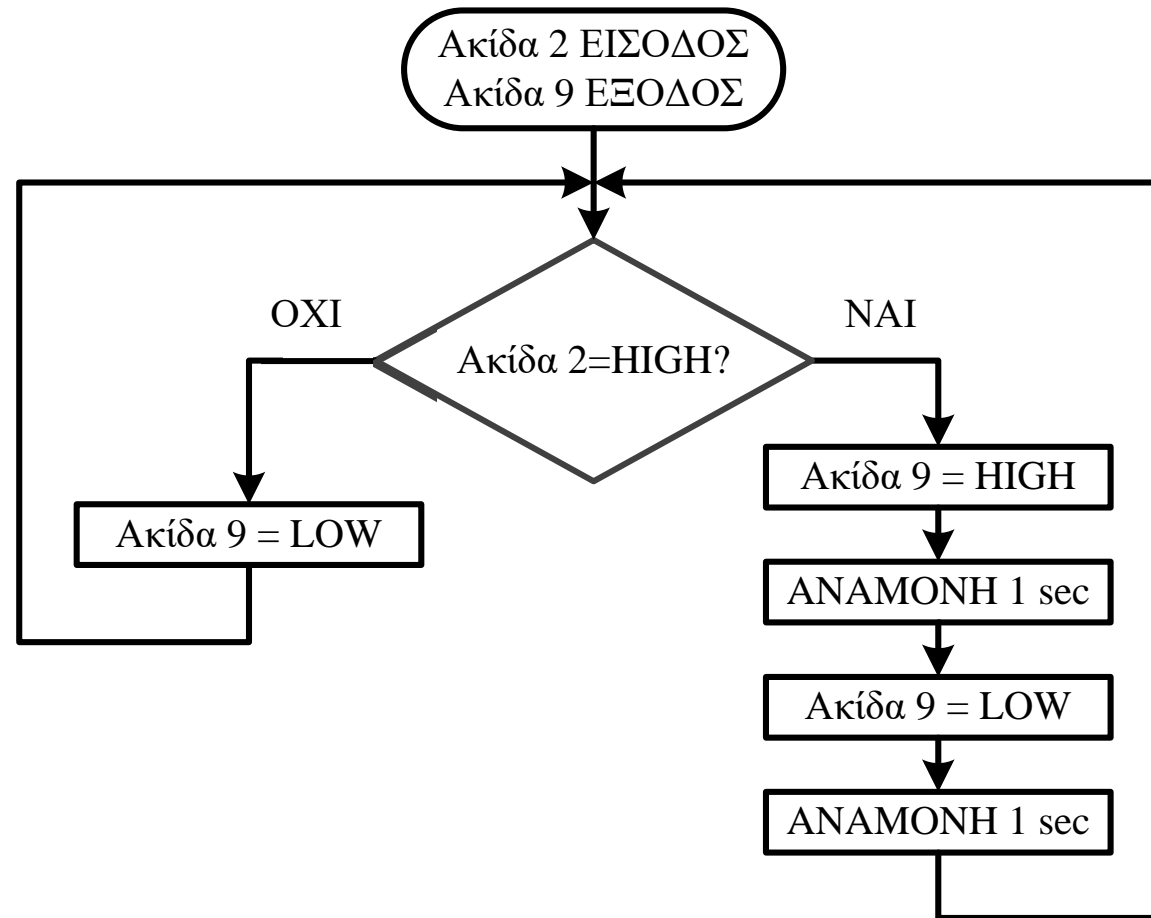
Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα αναβοσβήνει ή θα κρατά σβηστό ένα LED κάτω από τον έλεγχο ενός διακόπτη ώθησης (push button).



Pushing Led – 3



Pushing Led – 3 – Λογικό Διάγραμμα



Pushing Led – 3 – Wiring C

```
int pinButton = 2;           //Ονόμασε ακίδα 2 pinButton
int ledPin = 9;             //Ονόμασε ακίδα 9 ledPin
int stateButton = 0;       //Αρχικοποίησε stateButton

void setup()                //Συνάρτηση αρχικοποίησης
{
  pinMode(pinButton, INPUT); //Κάνε ακίδα 2 είσοδο
  pinMode(ledPin, OUTPUT);   //και ακίδα 9 έξοδο
}
```

Pushing Led – 3 – Wiring C

```
void loop()
{
stateButton = digitalRead(pinButton);
if (stateButton == 1)
{
digitalWrite(ledPin , HIGH);
delay(1000);
digitalWrite(ledPin , LOW );
delay(1000);
}
else
digitalWrite( ledPin, LOW );
}
```

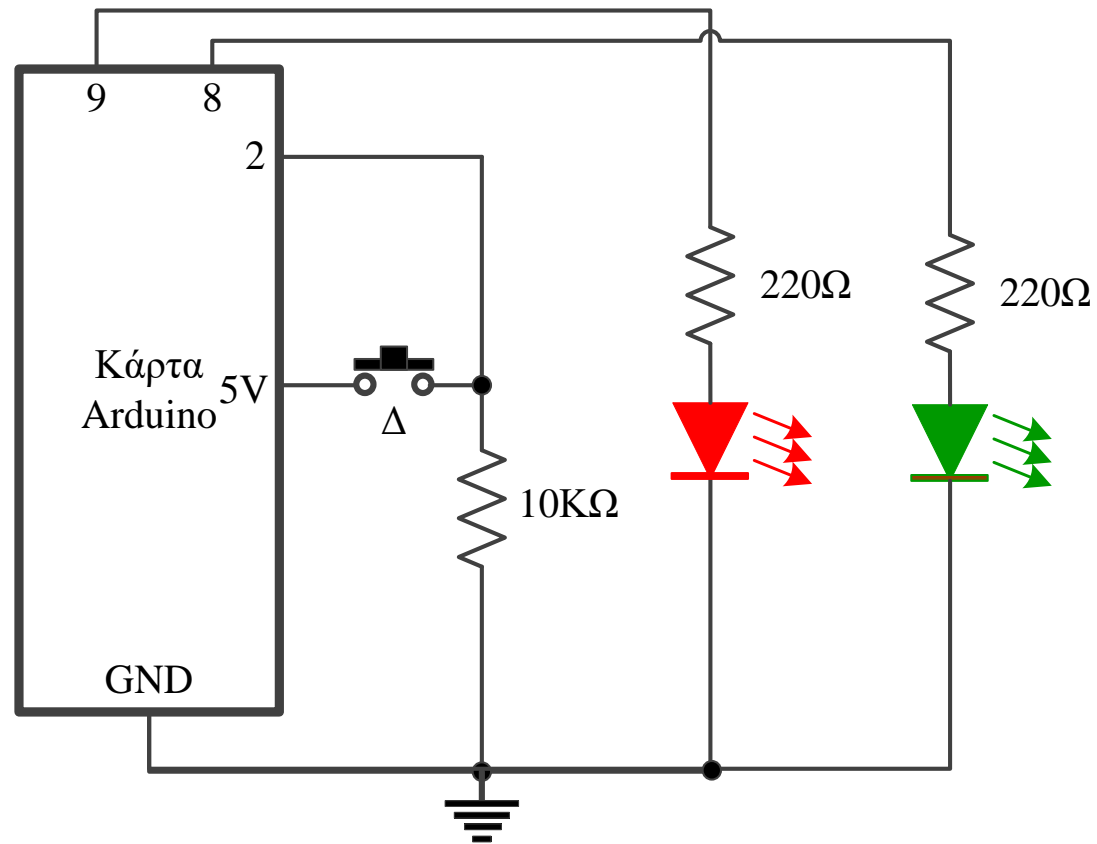
//Κύριος βρόχος
//Διάβασε την κατάσταση
//του διακόπτη και
//αν έχει κλείσει ο διακόπτης

//αναβόσβησε το LED

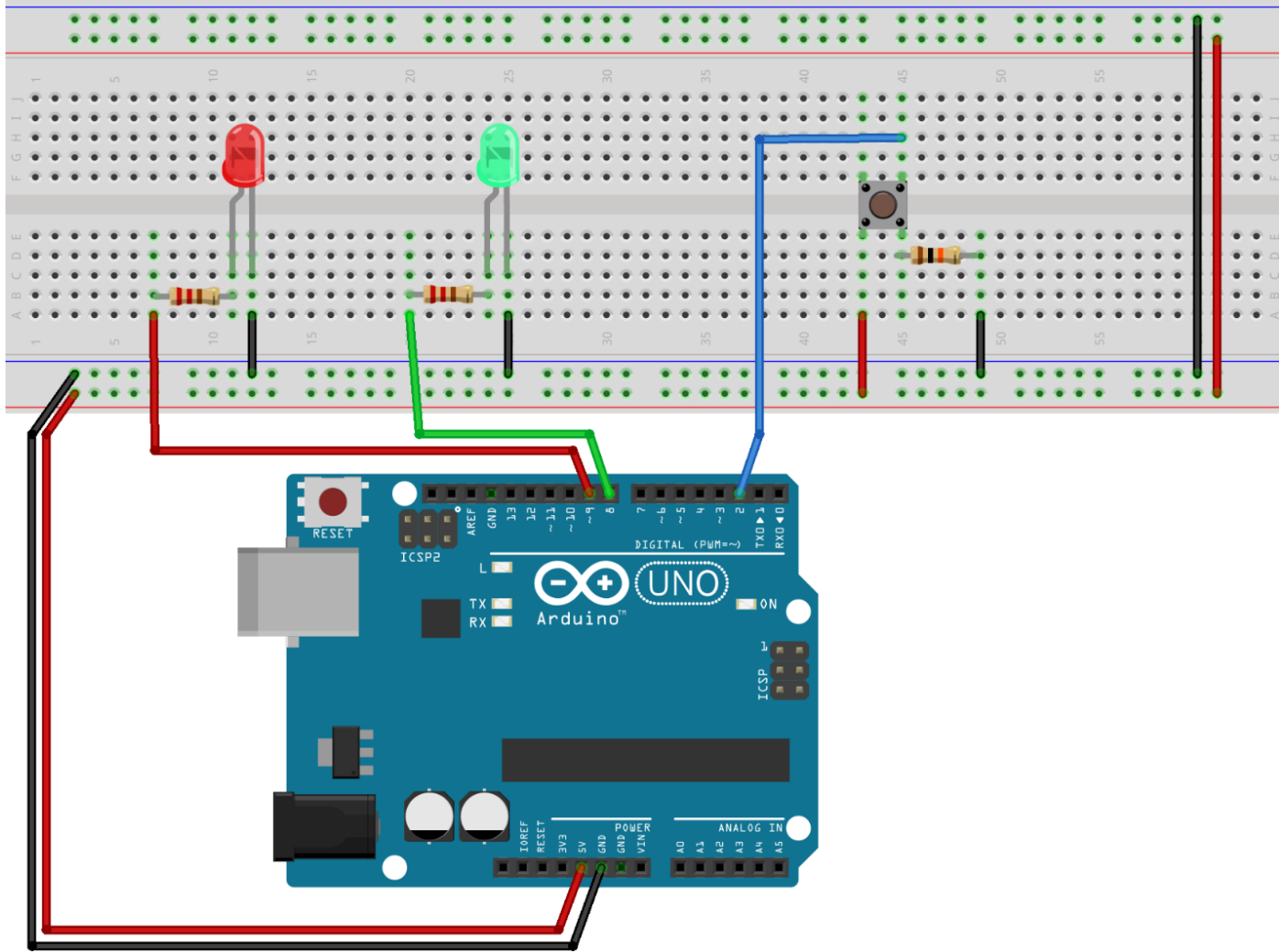
//διαφορετικά
//σβήσε το LED

Pushing Led – 4

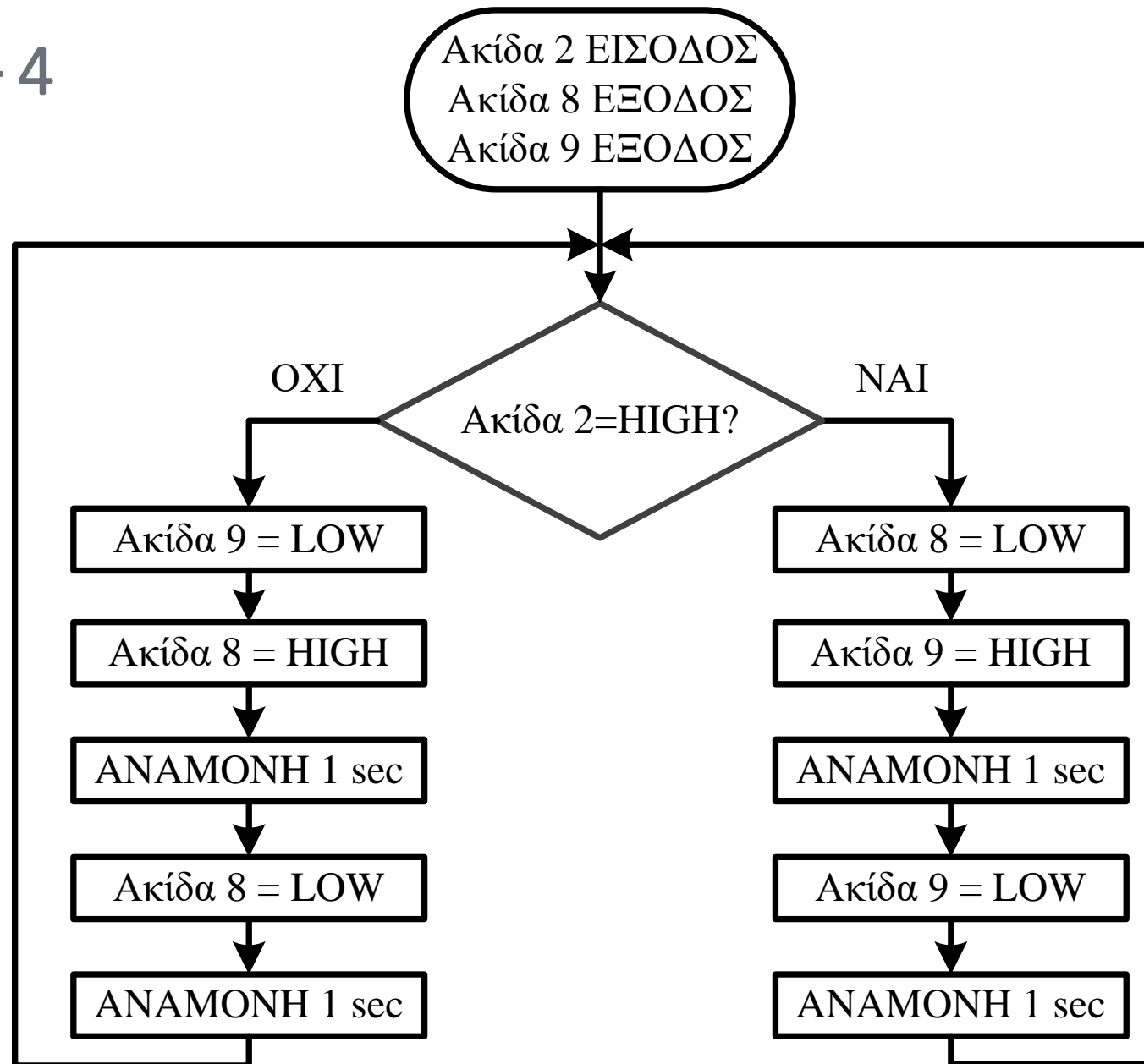
Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα αναβοσβήνει είτε ένα κόκκινο είτε ένα πράσινο LED κάτω από τον έλεγχο ενός διακόπτη ώθησης (push button).



Pushing Led – 4



Pushing Led – 4



Pushing Led – 4 – Wiring C

```
int buttonPin = 2;           //Ονόμασε ακίδα 2 buttonPin
int GledPin = 8;            //Ονόμασε ακίδα 8 GledPin
int RledPin = 9;           //Ονόμασε ακίδα 9 RledPin

void setup()
{
  pinMode( buttonpin, INPUT); //Κάνε την ακίδα 2 είσοδο
  pinMode( GledPin , OUTPUT); //και τις ακίδες 8 και 9
  pinMode( RledPin , OUTPUT); // εξόδους
}
```

Pushing Led – 4 – Wiring C

```
void loop()
{
  if (digitalRead(buttonPin)           //Αν η ακίδα 2 είναι HIGH
  {
    digitalWrite( RledPin, LOW );      //Αναβόσβησε πράσινο LED
    digitalWrite( GledPin , HIGH );
    delay( 1000 );
    digitalWrite( GledPin , LOW );
    delay( 1000 );
  }
  else                                   //Διαφορετικά
  {
    digitalWrite( GledPin , LOW );     //Αναβόσβησε κόκκινο LED
    digitalWrite( RledPin , HIGH );
    delay( 1000 );
    digitalWrite( RledPin , LOW );
    delay( 1000 );
  }
}
```

Pushing Led – 4 – Wiring C – Subroutines (version)

```
int buttonPin = 2;           //Ονόμασε ακίδα 2 buttonPin
int GledPin = 8;            //Ονόμασε ακίδα 8 GledPin
int RledPin = 9;           //Ονόμασε ακίδα 9 RledPin
int FlashTime = 500;       //Αρχική τιμή FlashTime=0
void FlashGreen();         //Υπορουτίνα αναβοσβήματος πράσινου LED
void FlashRed();           //Υπορουτίνα αναβοσβήματος κόκκινου LED
```

```
void setup()
{
  pinMode(buttonPin, INPUT);      //Κάνε την ακίδα 2 είσοδο
  pinMode(GledPin , OUTPUT);     //και τις ακίδες 8 και 9
  pinMode(RledPin , OUTPUT);     // εξόδους
}
```

Pushing Led – 4 – Wiring C – Subroutines (version)

```
void loop()                                //Κυρίως πρόγραμμα
{
    if (digitalRead( buttonPin ))          //Αν η ακίδα 2 είναι HIGH
    {                                       //αναβόσβησε πράσινο LED
        FlashGreen();
    }
    else                                    //Διαφορετικά
    {                                       //αναβόσβησε κόκκινο LED
        FlashRed();
    }
}
```

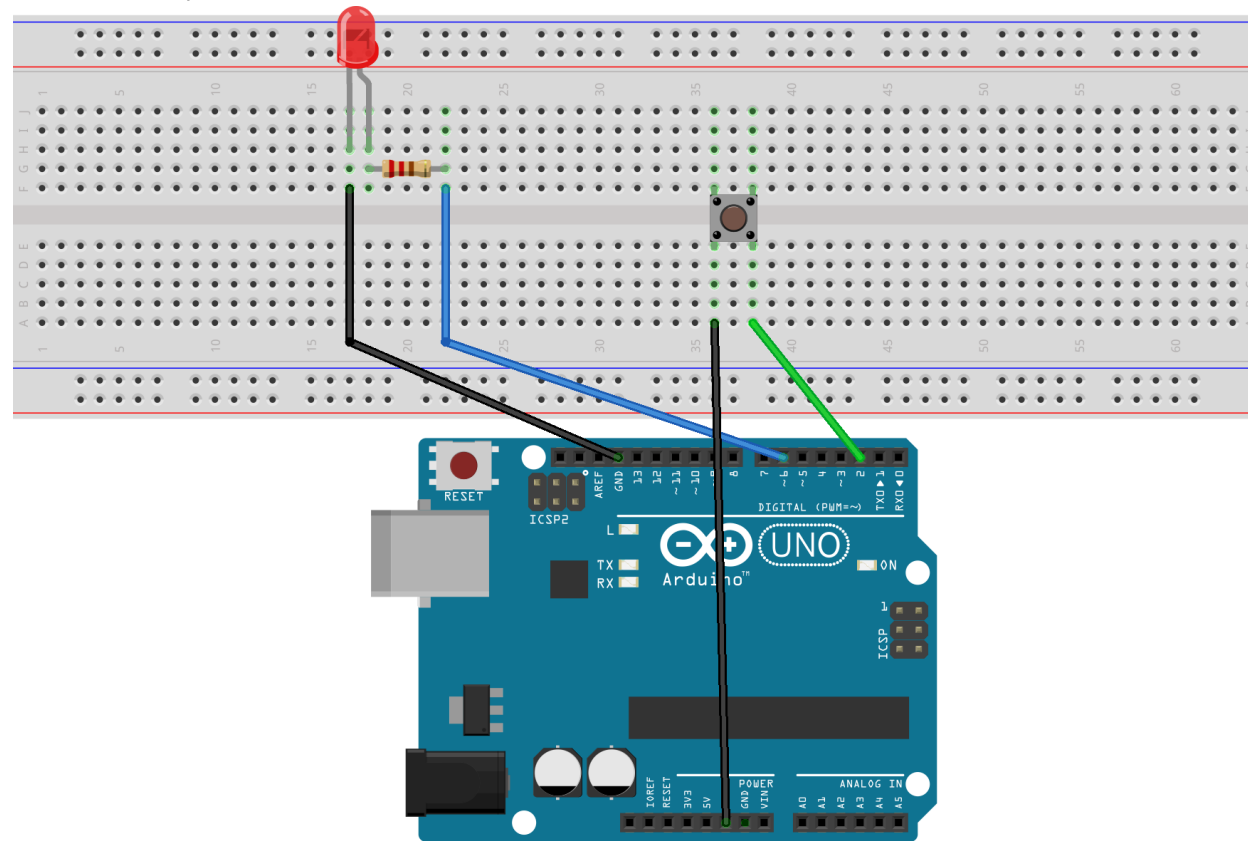
Pushing Led – 4 – Wiring C – Subroutines (version)

```
void FlashGreen() //Αναβόσβησε πράσινο LED
{ //με συχνότητα μία φορά
  digitalWrite(RledPin, LOW); //το δευτερόλεπτο
  digitalWrite(GledPin, HIGH);
  delay(FlashTime);
  digitalWrite(GledPin, LOW);
  delay(FlashTime);
}
```

```
void FlashRed() //Αναβόσβησε κόκκινο LED
{ //με συχνότητα μία φορά
  digitalWrite(GledPin, LOW); //το δευτερόλεπτο
  digitalWrite(RledPin, HIGH);
  delay(FlashTime);
  digitalWrite(RledPin, LOW);
  delay(FlashTime);
}
```

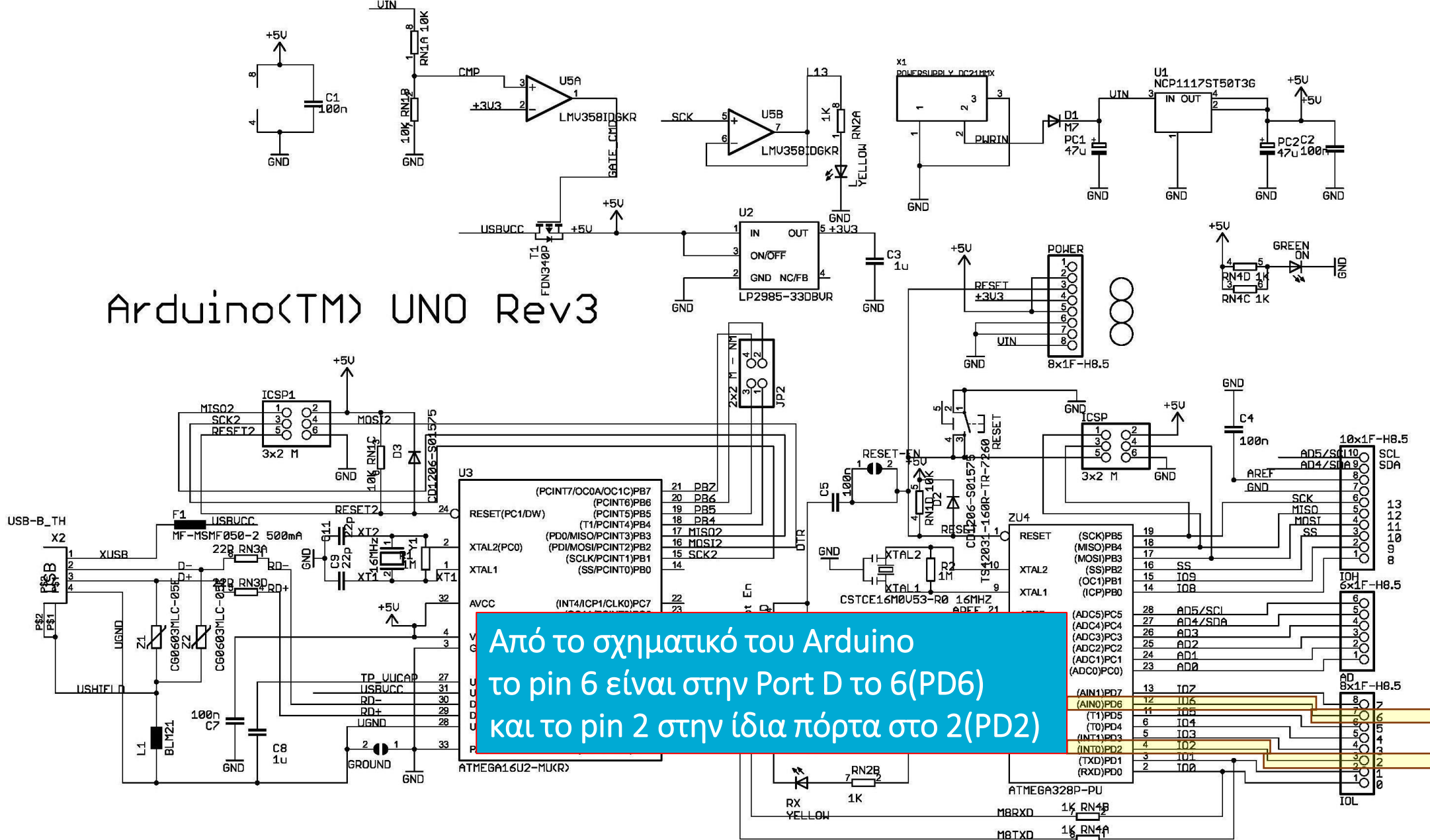
Pushing Led – 5 – SFR (Special Function Registers)

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino, Wiring C και τους SFR για να αναβοσβήνει ένα LED (PIN6) με ένα διακόπτη ώθησης (push button – PIN2) με ενεργοποιημένες τις Pull-up resistors στον διακόπτη.

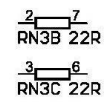


fritzing

Arduino(TM) UNO Rev3



Από το σχηματικό του Arduino το pin 6 είναι στην Port D το 6(PD6) και το pin 2 στην ίδια πόρτα στο 2(PD2)



Pushing Led – 5 – SFR (Special Function Registers)

```
int pinButton = 2;  
int ledPin = 6;
```

```
void setup(){  
  PORTD = B00000100;           //pinMode(pinButton, INPUT_PULLUP);  
  DDRD  = B01000000;           //pinMode(ledPin, OUTPUT);  
}
```

```
void loop(){  
  int stateButton = (PIND & (1 << pinButton)) >> pinButton; //int stateButton = digitalRead(pinButton);  
  
  if (stateButton == LOW) {  
    PORTD = (1 << ledPin) | PORTD;           //digitalWrite(ledPin, HIGH);  
  } else  
    PORTD = ~(1 << ledPin) & PORTD;         //digitalWrite(ledPin, LOW);  
}
```

Done Saving.

Sketch uses 462 bytes (1%) of program storage space. Maximum is 32256 bytes.

Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

Pushing Led – 5 – SFR (Special Function Registers)

76543210
DDRD = B01000000;

pinButton που βρίσκεται port D θέση 2 πρέπει να οριστεί ως **είσοδος** άρα **0**.
 ledPin που βρίσκεται port D θέση 6 πρέπει να οριστεί ως **έξοδος** άρα **1**.

76543210
PORTD = B00000100;

pinButton είναι είσοδος με ενεργοποιημένη την Pull-up άρα **1**.

PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

```
#define PIND_SFR_IO8 (0x09)
#define PIND0 0
#define PIND1 1
#define PIND2 2
#define PIND3 3
#define PIND4 4
#define PIND5 5
#define PIND6 6
#define PIND7 7
```

```
#define DDRD_SFR_IO8 (0x0A)
#define DDD0 0
#define DDD1 1
#define DDD2 2
#define DDD3 3
#define DDD4 4
#define DDD5 5
#define DDD6 6
#define DDD7 7
```

```
#define PORTD_SFR_IO8 (0x0B)
#define PORTD0 0
#define PORTD1 1
#define PORTD2 2
#define PORTD3 3
#define PORTD4 4
#define PORTD5 5
#define PORTD6 6
#define PORTD7 7
```

DDRx, PORTx, PINx είναι macros στο Arduino (iom328p.h)



Pushing Led – 5 – SFR (Special Function Registers)

76543210
DDRD = B0**1**000**0**00;
76543210
PORTD = B00000**1**00;

pinButton που βρίσκεται port D θέση 2 πρέπει να οριστεί ως **είσοδος** άρα **0**.
ledPin που βρίσκεται port D θέση 6 πρέπει να οριστεί ως **έξοδος** άρα **1**.
pinButton είναι είσοδος με ενεργοποιημένη την Pull-up άρα **1**.

`int stateButton = digitalRead(pinButton);` με τις SFR δηλώσεις:

`int stateButton = (PIND & (1 << pinButton)) >> pinButton;`

Εδώ θα χρησιμοποιήσουμε το PINx γιατί αυτό αποθηκεύει την τιμή του κάθε bit του SFR. Εμείς στο pin εισόδου θέλουμε να βλέπουμε την τιμή του για το ανάβουμε ή να το σβήνουμε (πατημένο ανάβει/ελεύθερο είναι σβηστό). Εδώ χρειαζόμαστε δύο ολισθήσεις, η πρώτη προς τα αριστερά για να απομονώσουμε μόνο την τιμή της εισόδου και η δεύτερη προς τα δεξιά για να φέρουμε την τιμή 0 ή 1 στο bit 0.

76543210
PIND = B00000**x**00
&AND MASK= B00000**1**00 Μάσκα με LSL κατά pinButton =2 θέσεις τον 1 (1 << pinButton)

PIND = B00000**x**00 και ολίσθηση προς τα δεξιά LSR κατά pinButton θέσεις (>> pinButton).

PIND = B0000000**x** το x μπορεί να είναι 0 ή 1 ανάλογα με το πάτημα του button και μπορεί να γίνει ο έλεγχος του if (stateButton == LOW).

Pushing Led – 5 – SFR (Special Function Registers)

76543210
DDRD = B0**1**000**0**00;
pinButton που βρίσκεται port D θέση 2 πρέπει να οριστεί ως **είσοδος** άρα **0**.
ledPin που βρίσκεται port D θέση 6 πρέπει να οριστεί ως **έξοδος** άρα **1**.

76543210
PORTD = B00000**1**00;
pinButton είναι είσοδος με ενεργοποιημένη την Pull-up άρα **1**.

digitalWrite(ledPin, HIGH); με τις SFR δηλώσεις:

```
if (stateButton == LOW) {  
    PORTD = (1 << ledPin) | PORTD;   ή αρκεί....   PORTD = B01000000; ???  
}
```

Στη δεύτερη δήλωση θα δίναμε τιμή 1 στο pin 6 αλλά θα χάναμε την ενεργοποίηση της εισόδου στο 2, άρα πρέπει να χρησιμοποιήσουμε μια μάσκα με OR ώστε να μείνουν και τα δύο:

76543210
PORTD = B00000**1**00

|OR MASK = B0**1**000000 Μάσκα με LSL κατά ledPin=6 θέσεις τον 1 (1 << ledPin)

PORTD = B0**1**000**1**00 Άρα έχουμε και το pin 2 ανεπηρέαστο και το pin 6 set στο 1.

Pushing Led – 5 – SFR (Special Function Registers)

76543210
DDRD = B0**1**000**0**00;
pinButton που βρίσκεται port D θέση 2 πρέπει να οριστεί ως **είσοδος** άρα **0**.
ledPin που βρίσκεται port D θέση 6 πρέπει να οριστεί ως **έξοδος** άρα **1**.

76543210
PORTD = B00000**1**00;
pinButton είναι είσοδος με ενεργοποιημένη την Pull-up άρα **1**.
else
digitalWrite(ledPin, LOW); **με τις SFR δηλώσεις:**

PORTD = ~(1 << ledPin) & PORTD; ή αρκεί.... PORTD = B00000000; **???**

Στη δεύτερη δήλωση δίναμε τιμή 0 στο pin 6 αλλά θα χάναμε την ενεργοποίηση της εισόδου στο 2, άρα πρέπει να χρησιμοποιήσουμε μια νέα μάσκα με not AND:

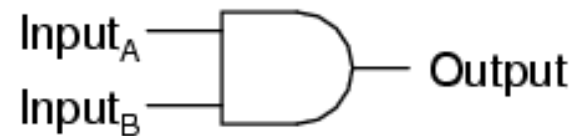
76543210
PORTD = B00000**1**00
& AND MASK = B**1**0**1**11111 Μάσκα με LSL κατά ledPin=6 θέσεις τον 1 και not: ~(1 << ledPin)

PORTD = B0**0**000**1**00 Άρα έχουμε και το pin 2 ανεπηρέαστο και το pin 6 set στο 0.

Logic Gate

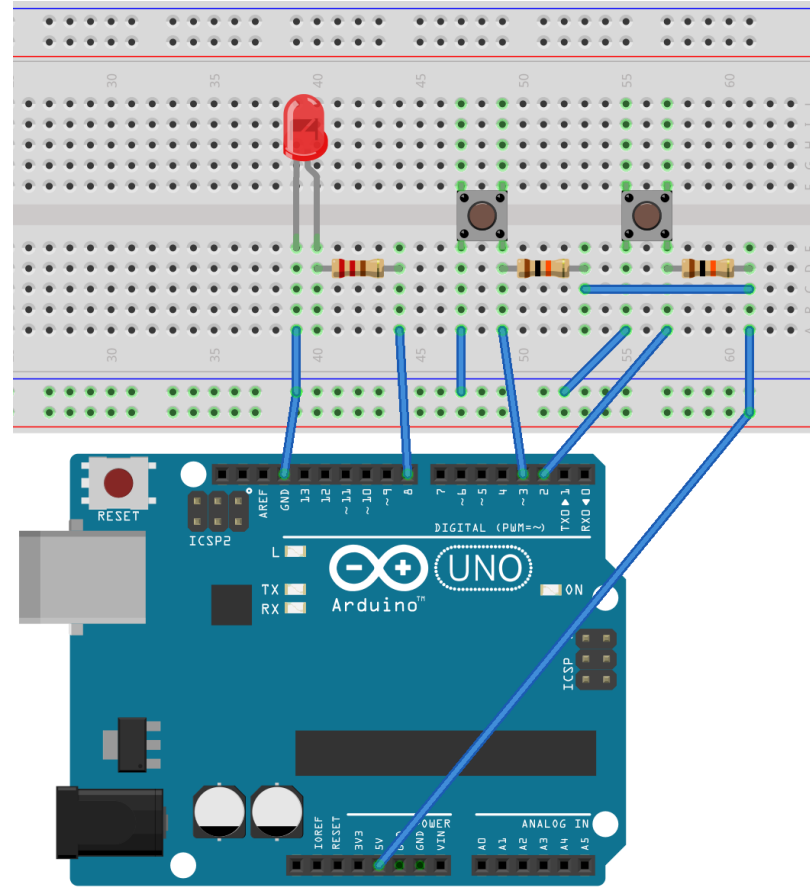
Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και θα υλοποιεί μια πύλη AND δύο εισόδων.

2-input AND gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Logic Gate



fritzing

Logic Gate – Coding

```
int buttonPinA = 2; // Button 1  
int buttonPinB = 3; // Button 2  
int LEDredF = 8; // Output pin Led
```

```
int A; // variable for input state of the Button 1  
int B; // variable for input state of the Button 2
```

```
void setup() {  
  pinMode(LEDredF, OUTPUT); // set led as output  
  pinMode(buttonPinA, INPUT); //set logic input A  
  pinMode(buttonPinB, INPUT); //set logic input B  
}
```

Logic Gate – Coding

```
void loop(){
```

```
A = digitalRead(buttonPinA);
```

```
B = digitalRead(buttonPinB);
```

```
if ((A&&B)) { // put here your logic statement, be careful with ()
```

```
digitalWrite(LEDredF, HIGH);
```

```
} else
```

```
{
```

```
digitalWrite(LEDredF, LOW);
```

```
}
```

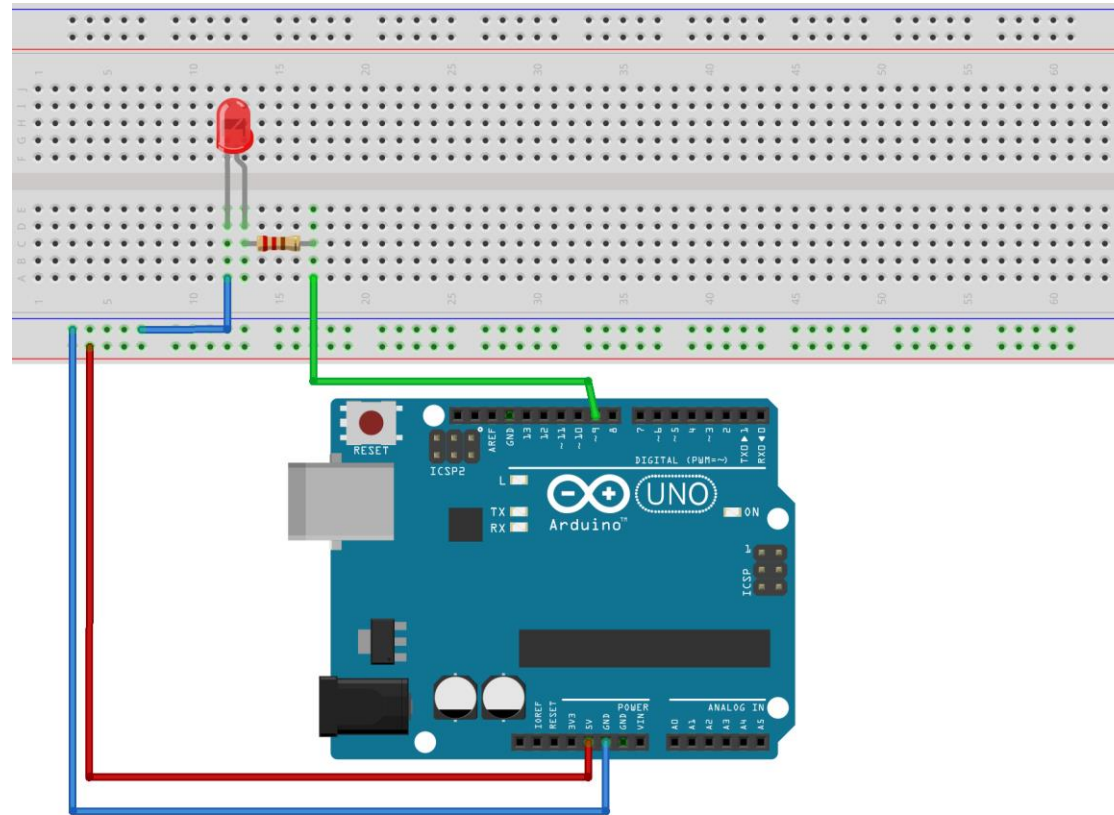
```
}
```

Με το ίδιο τρόπο υλοποιούμε και τις άλλες λογικές πύλες:

! (logical not) && (logical and) || (logical or)

Serial Monitor – Led Control

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα δέχεται από τη σειριακή αριθμούς από το 0 έως το 9. Για κάθε αριθμό 1 θα ανάβει ένα LED και με το 0 θα σβήνει. Για κάθε άλλο αριθμό η σειριακή θα ενημερώνει ότι δόθηκε λάθος αριθμός και η ενέργεια ήταν άκυρη.



fritzing

Serial Monitor – Led Control

```
const int ledPin = 9;
int number = 0;

void setup() {
    // initialize the serial communication:
    // Set also the monitor to 9600 bps
    //a short delay to establish serial communication
    // initialize the ledPin as an output:
    pinMode(ledPin, OUTPUT);
    // Change Serial Monitor line ending :
    Serial.println("Set monitor to No line ending");
    Serial.println("Enter Number:");
}
```

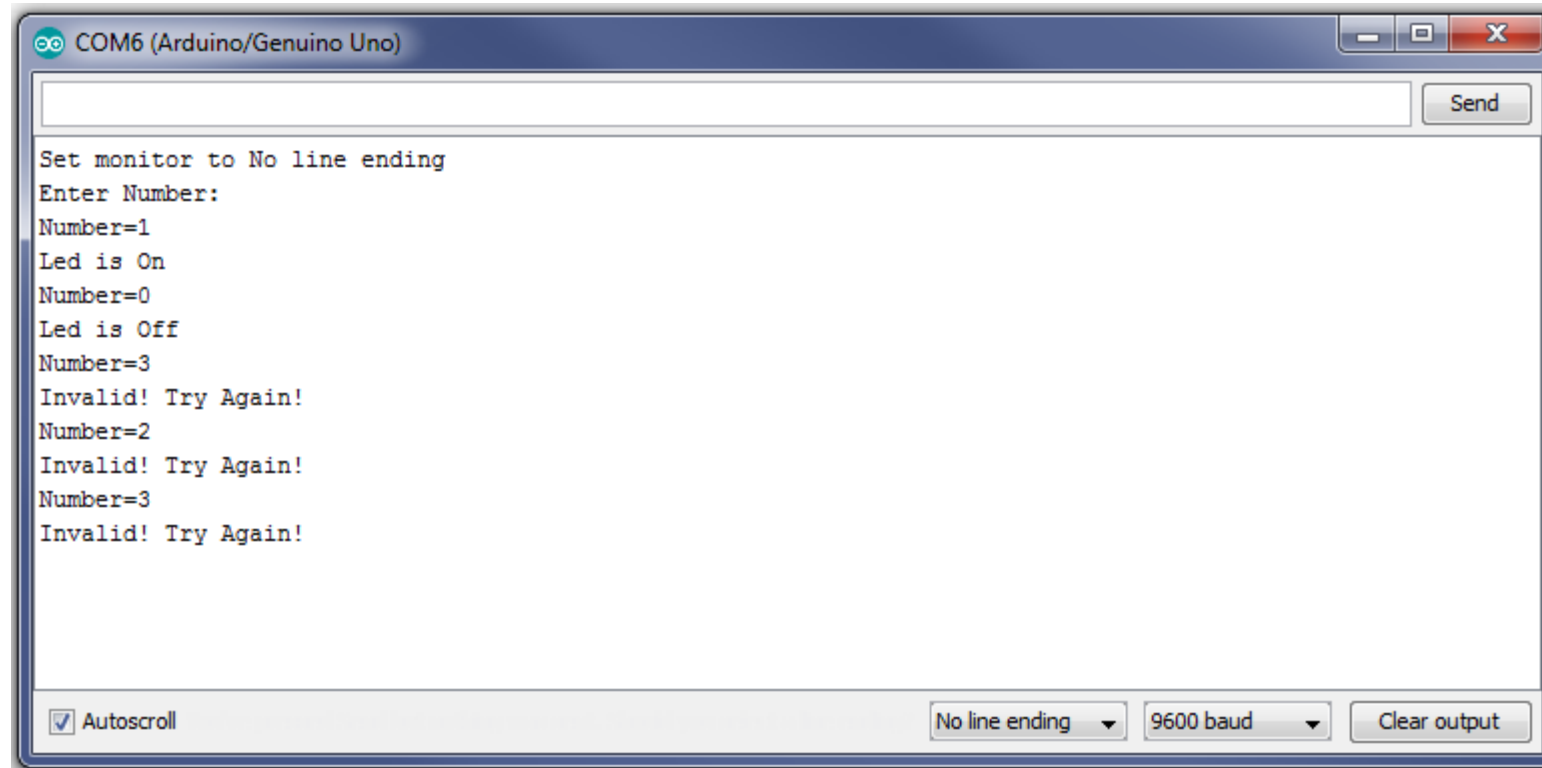
Serial Monitor – Led Control

```
void loop() {  
  number=0; // set number to 0 for each loop  
  while (Serial.available() == 0) { /*do nothing*/ }  
  
  while (Serial.available()) {  
    //Arduino reads from serial a byte/loop in ASCII coding  
    //byte incoming = Serial.read();  
    byte incoming = Serial.read() - '0';  
    //Corrects ASCII Dec to an integer digit from 0 to 9  
  
    number = incoming;  
    Serial.print("Number=");  
    Serial.println(number);  
  }  
}
```

Serial Monitor – Led Control

```
if ((number >=0 && number <=9) && (number == 1)){  
    Serial.println("Led is On");  
    digitalWrite(ledPin, HIGH);  
}  
else if ((number >=0 && number <=9) && (number == 0)) {  
    Serial.println("Led is Off");  
    digitalWrite(ledPin, LOW);  
}  
else {  
    Serial.println("Invalid! Try Again!");  
}  
} // close while  
} //close Loop
```

Serial Monitor – Led Control



Serial Monitor – A²

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα δέχεται από τη σειριακή ένα μη προσημασμένο ακέραιο αριθμό και θα του υπολογίζει τη δύναμη του 2.

Το αποτέλεσμα θα πρέπει να είναι ένας έγκυρος μη προσημασμένος ακέραιος.

Στην περίπτωση που έχουμε υπερχείλιση θα ανάβει ένα LED.

Serial Monitor – A^2

```
const int ledPin = 9;
unsigned int RESULT;
unsigned int A = 0;

void setup() {
    // initialize the serial communication:
    Serial.begin(9600);
    delay(50);           //give time to initialize serial
                       // initialize the ledPin as an output:
    pinMode(ledPin, OUTPUT);
    Serial.println("Set monitor to No line ending");
}
```

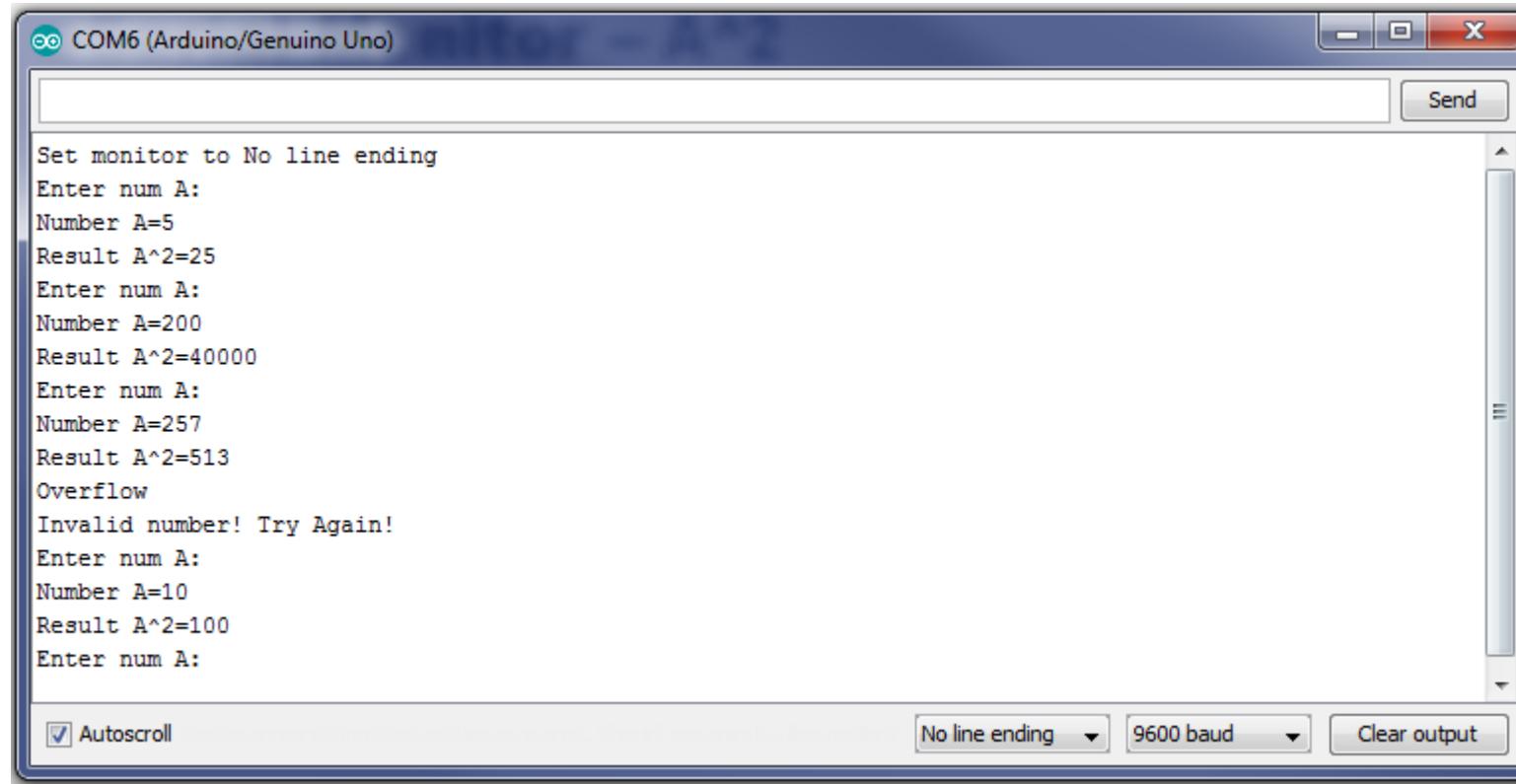
Serial Monitor – A²

```
void loop() {  
  A = 0;  
  Serial.println("Enter num A:");  
  while (Serial.available() == 0) { /*do nothing*/ }  
  
  while (Serial.available()) {  
  
    A = Serial.parseInt();  
    Serial.print("Number A=");  
    Serial.println(A);  
    RESULT = A*A;  
    Serial.print("Result A^2=");  
    Serial.println(RESULT);  
  }  
}
```

Serial Monitor – A^2

```
if (A >= 256) {  
    digitalWrite(ledPin, HIGH);  
    Serial.println("Overflow");                //OVRFLOW = 65535  
    Serial.println("Invalid number! Try Again!");  
}  
else {  
    digitalWrite(ledPin, LOW);  
}  
Serial.flush();  
                //calling flush() you can be sure that all data has been sent,  
                //and the buffer is empty  
}
```

Serial Monitor – A²



```
COM6 (Arduino/Genuino Uno)
Set monitor to No line ending
Enter num A:
Number A=5
Result A^2=25
Enter num A:
Number A=200
Result A^2=40000
Enter num A:
Number A=257
Result A^2=513
Overflow
Invalid number! Try Again!
Enter num A:
Number A=10
Result A^2=100
Enter num A:
```

Autoscroll No line ending 9600 baud Clear output

Arduino Analog Pins

Το Arduino προσφέρει ένα σύνολο ακροδεκτών που συνδέονται απευθείας με τις αναλογικές εισόδους του μικροελεγκτή. Το αναλογικό σήμα εισόδου μπορεί να βρίσκεται στο εύρος 0-5V.

Ο μικροελεγκτής μετατρέπει τις αναλογικές τάσεις έτσι ώστε να αντιστοιχούν σε **1024** διαφορετικές ψηφιακές στάθμες με τη χρήση του ενσωματωμένου 10-bit Μετατροπέα από Αναλογικό σε Ψηφιακό (10-bit Analog to Digital Converter – $2^{10}=1024$). Ανάλογα με τη συσκευασία του έχει 6-κανάλια (PDIP) και 8-κανάλια (TQFP, QFN/MLF). **Άρα διαθέτει ακρίβεια μετατροπής $5V/1024=0.0049V$ ή $4.9mV$.**

Η ακρίβεια αυτή είναι ικανοποιητική για τις περισσότερες εφαρμογές.

Η ανάγνωση αναλογικού σήματος γίνεται με την εντολή:

analogRead(pin) όπου pin το αναγνωριστικό της αναλογικής εισόδου.

Οι δύο επόμενες εντολές είναι ισοδύναμες και αναφέρονται στην ανάγνωση από την αναλογική είσοδο 0:

analogRead(0)

analogRead(A0)

Arduino Analog Pins

Το πρόβλημα που προκύπτει είναι όταν το εύρος των αναλογικών τάσεων είναι περιορισμένο, π.χ. 0-3V.

Σε μια τέτοια περίπτωση η ανάλυση που προκύπτει θα είναι πάλι η ίδια!, δηλαδή 0.0049.

Αυτό συμβαίνει διότι ο μικροελεγκτής χρησιμοποιεί εξ' ορισμού ως τάση αναφοράς τα 5V.

Το ζητούμενο όμως είναι η βελτίωση της ανάλυσης.

Για να γίνει αυτό, θα πρέπει να προσαρμοστεί η τάση αναφοράς σε μια κοντινή τιμή με τη μέγιστη της αναλογικής εισόδου που θα εφαρμοστεί.

Η ρύθμιση αυτή μπορεί να γίνει είτε προγραμματιστικά είτε με εφαρμογή εξωτερικού κυκλώματος.

Στην προγραμματιστική ρύθμιση, οι επιλογές είναι περιορισμένες σε αντίθεση με το προσαρμοσμένο εξωτερικό κύκλωμα που μπορεί να χρησιμοποιηθεί.

Arduino Analog Pins

Η ρύθμιση της τάσης αναφοράς γίνεται με την εντολή `analogReference(Ref_type)`, όπου η παράμετρος `Ref_type` μπορεί να είναι:

DEFAULT

Αντιστοιχεί σε τάση 5V ή 3.3V ανάλογα με την έκδοση του Arduino.

INTERNAL

Στις περισσότερες περιπτώσεις (ATmega168 or ATmega328P) αντιστοιχεί στην τιμή 1.1V (2.56V στον μικροελεγκτή ATmega8).

Στην έκδοση Mega χρησιμοποιούνται οι INTERNAL1V1 και INTERNAL2V56 αντί της INTERNAL για τάσεις αναφοράς 1.1V και 2.56V αντίστοιχα.

Arduino AREF (EXTERNAL)

Με αυτή την παράμετρο, η τάση αναφοράς ρυθμίζεται από την τάση που εφαρμόζεται στον ακροδέκτη **AREF**.

Πάντως, αυτή η τάση δεν μπορεί να είναι μικρότερη από 0V ή μεγαλύτερη από 5V.

Η τάση αυτή μπορεί να προσαρμοστεί εύκολα στο επιθυμητό επίπεδο χρησιμοποιώντας ένα διαιρέτη τάσης.

Έτσι, με τάση αναφοράς τα 3V, η **αντίστοιχη ανάλυση θα είναι $3/1024=0.003V$** , δηλαδή πολύ καλύτερη από την 0.0049V στην περίπτωση της τάσης αναφοράς των 5V.

Με αυτό τον τρόπο επιτυγχάνεται μεγαλύτερη ευχέρεια διαχείρισης ενός αναλογικού σήματος που κινείται στην περιοχή 0 έως 3V.

analogRead()



Reads the value from the specified analog pin. The Arduino board contains a 6 channel A0-A5 (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter.

- This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023.
- This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit.
- The input range and resolution can be changed using [analogReference\(\)](#).
- It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

analogReference()

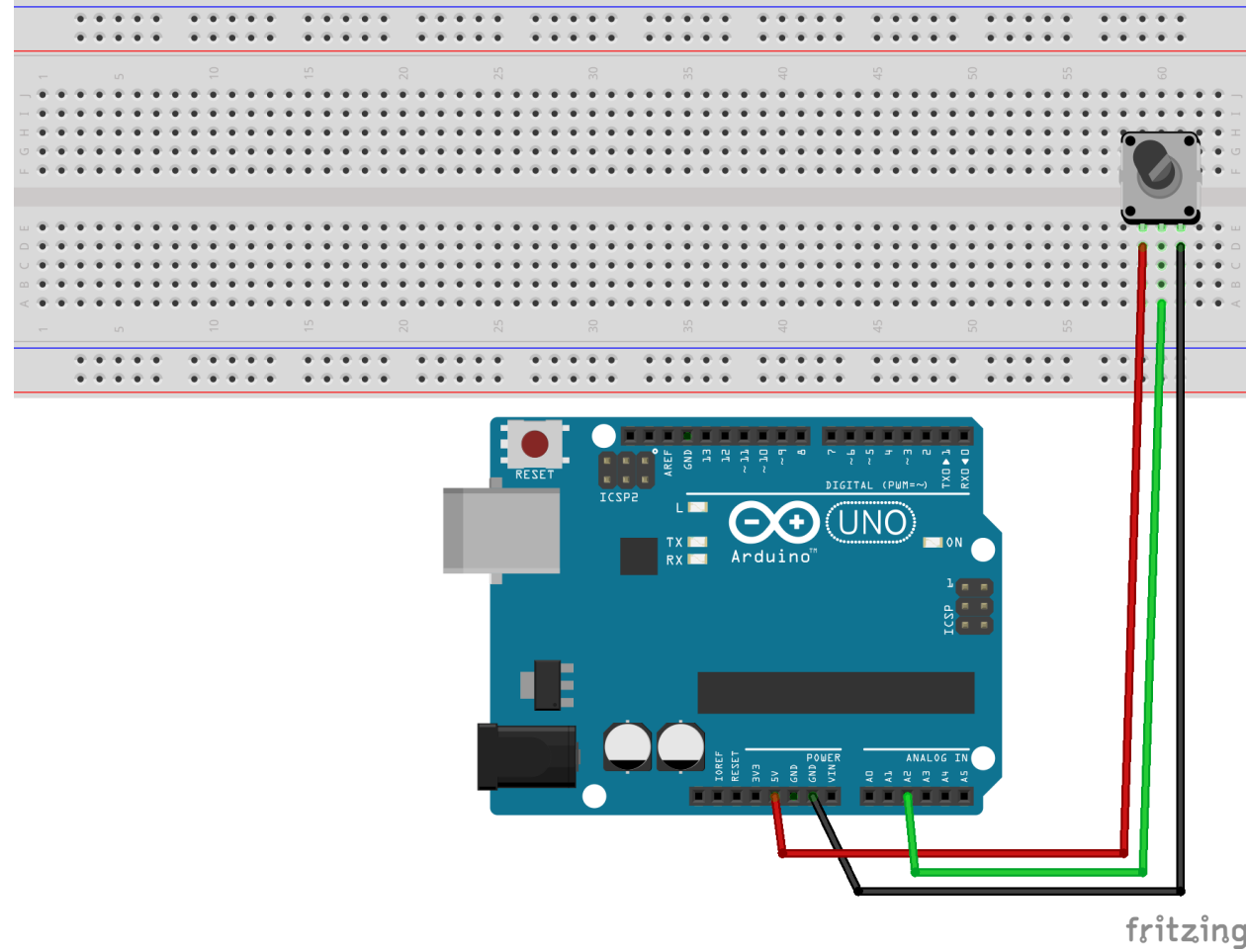


Notes and Warnings!!!

After changing the analog reference, the first few readings from `analogRead()` may not be accurate. Don't use anything less than 0V or more than 5V for external reference voltage on the AREF pin! If you're using an external reference on the AREF pin, you must set the analog reference to EXTERNAL before calling `analogRead()`. Otherwise, you will short together the active reference voltage (internally generated) and the AREF pin, possibly damaging the microcontroller on your Arduino board.

Alternatively, you can connect the external reference voltage to the AREF pin through a 5K resistor, allowing you to switch between external and internal reference voltages. Note that the resistor will alter the voltage that gets used as the reference because there is an internal 32K resistor on the AREF pin. The two act as a voltage divider, so, for example, 2.5V applied through the resistor will yield $2.5 * 32 / (32 + 5) = \sim 2.2V$ at the AREF pin.

Debug analogPin



Debug analogPin

```
int analogPin = 2;           // potentiometer wiper (middle terminal)
                              //connected to analog pin 2
                              // outside leads to ground and +5V
int val = 0;                 // variable to store the value read

void setup()
{
  Serial.begin(9600);        // setup serial
}

void loop()
{
  val = analogRead(analogPin); // read the input pin
  Serial.println(val);         // debug value
}
```

Σημείωση: Οι ακροδέκτες GPIO με την ένδειξη Ax, χρησιμοποιούνται κατά βάση ως αναλογικοί (δεν δηλώνονται στο setup) αλλά μη ξεχνάμε ότι είναι IO γενικού σκοπού και μπορούν να δηλωθούν στο setup ως ψηφιακές.

Περισσότερα εδώ:

<https://docs.arduino.cc/learn/microcontrollers/analog-input>

PWM – Pulse Width Modulation

Η **τεχνική PWM** αποτελεί και για το Arduino μια μέθοδο παραγωγής ψευδοαναλογικών τάσεων μέσω των ψηφιακών εξόδων.

Όλες οι εκδόσεις Arduino, υποστηρίζουν PWM στους ψηφιακούς ακροδέκτες.

Οι ακροδέκτες που μπορούν να δώσουν τέτοιο σήμα, έχουν την **ένδειξη (~)**.

Στο Arduino UNO διαθέτουν PWM οι ακίδες: **3,5,6,9,10,11**

Η τεχνική αυτή λύνει το πρόβλημα των ενδιάμεσων τάσεων που απαιτούν για παράδειγμα τα μοτέρ DC προκειμένου να είναι δυνατή η επιλογή της ταχύτητας περιστροφής ή το ομαλό άναμμα και σβήσιμο LED (fading).

PWM – Duty Cycle – analogWrite

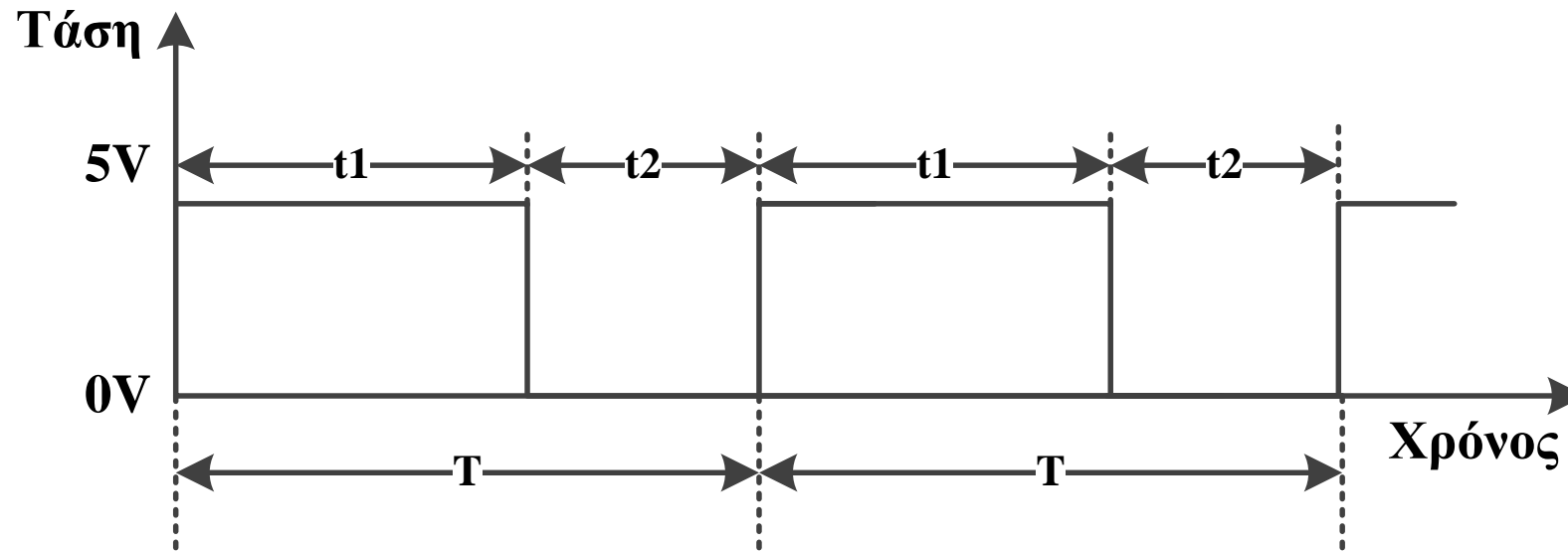
Ο καθορισμός του ωφέλιμου κύκλου (Duty Cycle) γίνεται με την εντολή **analogWrite**, που παίρνει ως παράμετρο έναν ακέραιο αριθμό στο διάστημα 0 έως 255.

Ο αριθμός 0 αντιστοιχεί σε ποσοστό ωφέλιμου κύκλου 0%, ενώ ο 255 σε ποσοστό 100%.

Οι εντολές που ακολουθούν, ρυθμίζουν τα αντίστοιχα ποσοστά για τα PWM σήματα που φαίνονται στο επόμενο σχήμα για τον ακροδέκτη pin.

```
analogWrite(Pin, 0)           // DC=0%
analogWrite(Pin, 63)          // DC=25%
analogWrite(Pin, 127)         // DC=50%
analogWrite(Pin, 191)         // DC=75%
analogWrite(Pin, 255)         // DC=100%
```

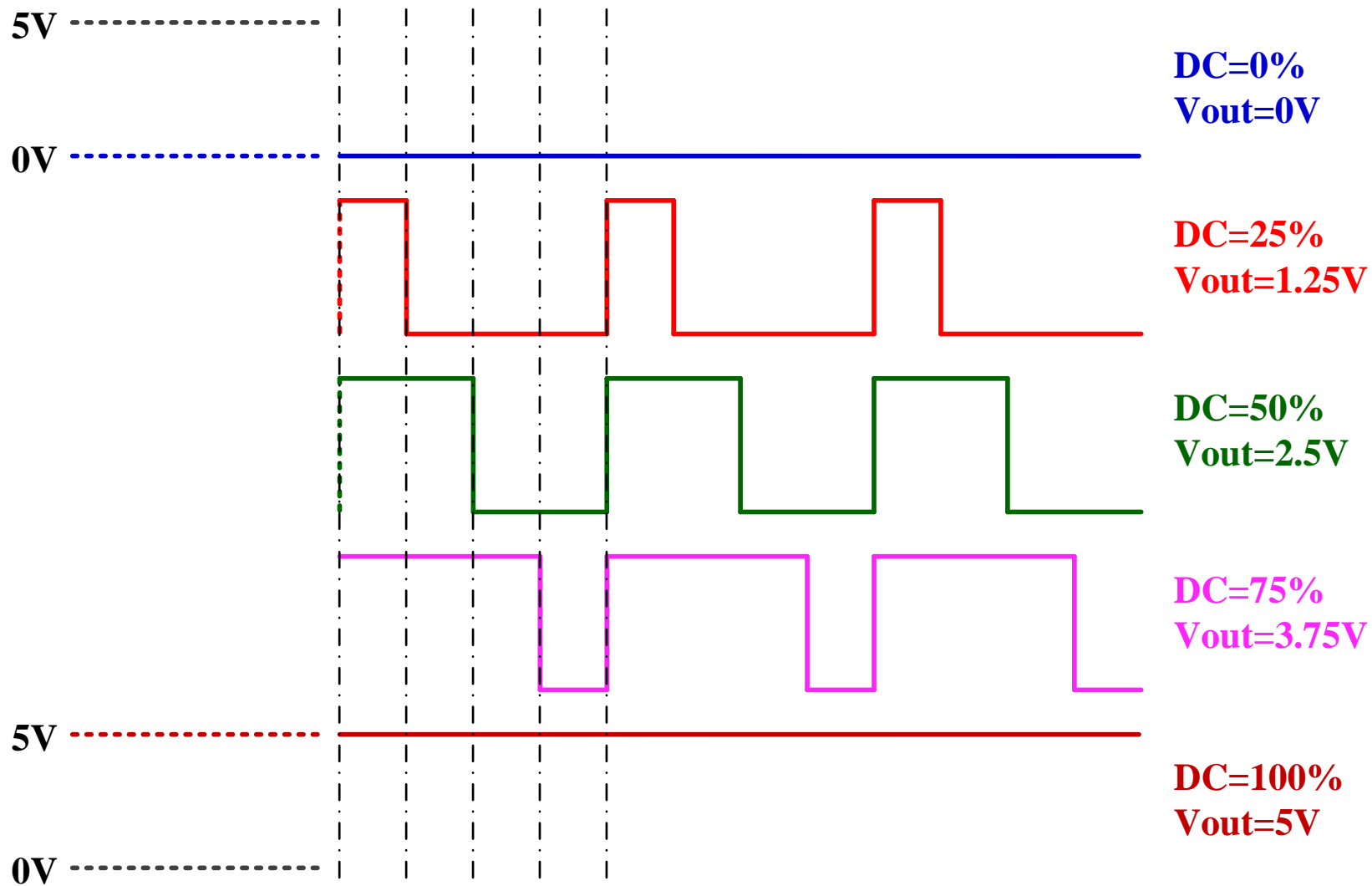
Δημιουργία Παλμών Διαμόρφωσης Εύρους Παλμού PWM (Pulse Width Modulation)



Κύκλος ενεργού λειτουργίας (Duty Cycle ή DC) = $t1/T(\%)$

Το Arduino έχει PWM συχνότητα $f \sim 500\text{Hz}$ άρα η περίοδος $T=2\text{ms}$.
Μια κλήση στην `analogWrite()` μπορεί να πάρει τιμές από 0 – 255.
Άρα, π.χ. μια `analogWrite(255)` έχει 100% duty cycle (αναμμένο μέγιστα),
και `analogWrite(127)` είναι 50% duty cycle (αναμμένο μέτρια).

Ενδεικτικές κυματομορφές για την τεχνική PWM



analogWrite()



Writes an analog value (PWM wave) to a pin. After a call to `analogWrite()`, the pin will generate a steady square wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.

- The frequency of the PWM signal on most pins is approximately 490 Hz.
- On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.
- On most Arduino boards (those with the ATmega168 or ATmega328P), this function works on pins 3, 5, 6, 9, 10, and 11.
- You do not need to call `pinMode()` to set the pin as an output before calling `analogWrite()`.
- The `analogWrite` function has nothing to do with the analog pins or the `analogRead` function.

map()



Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

Does not constrain values to within the range, because out-of-range values are sometimes intended and useful. The constrain() function may be used either before or after this function, if limits to the ranges are desired.

Note that the "lower bounds" of either range may be larger or smaller than the "upper bounds" so the map() function may be used to reverse a range of numbers, for example: **y = map(x, 1, 50, 50, 1);**

The function also handles negative numbers well, so that this example: **y = map(x, 1, 50, 50, -100);** is also valid and works well.

The map() function uses integer math so will not generate fractions, when the math might indicate that it should do so. Fractional remainders are truncated and are not rounded or averaged.

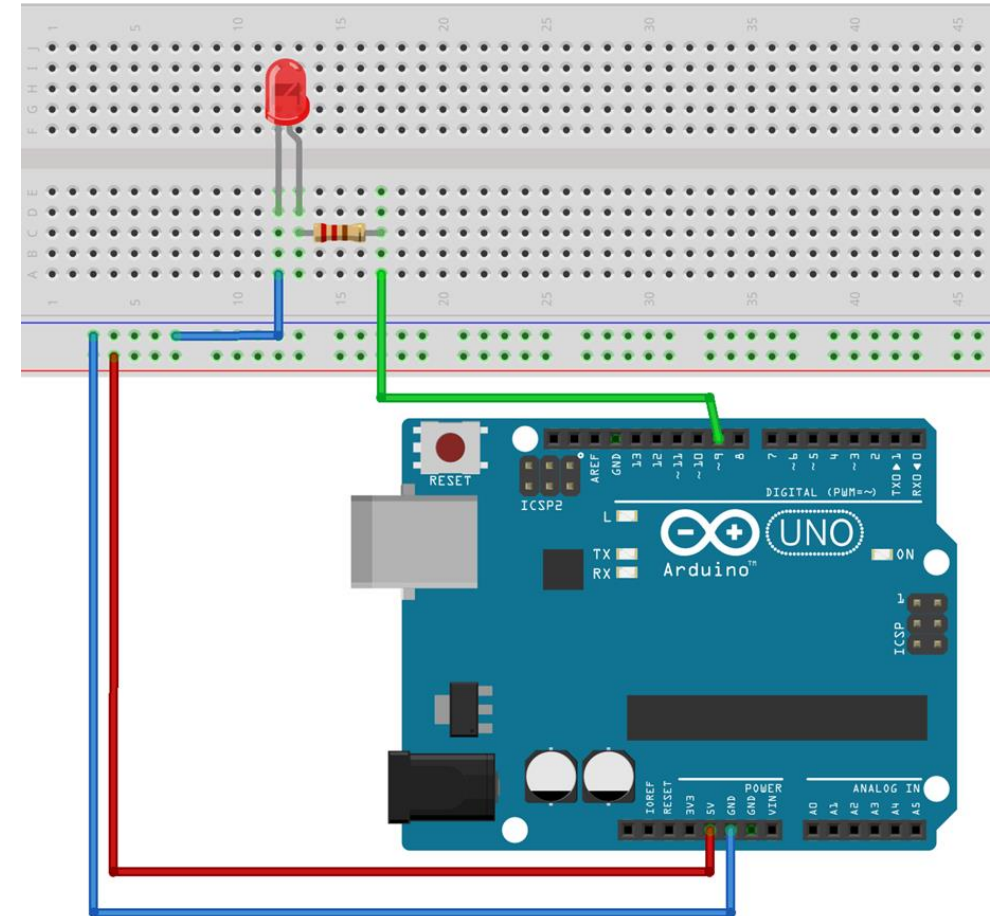
Syntax

map(value, fromLow, fromHigh, toLow, toHigh)



Εφαρμογές με PWM – 1

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα αυξάνει την φωτεινότητα ενός LED με τη χρήση PWM από 0 έως 255.



Notes and Warnings!!!

The PWM outputs generated on pins 5 and 6 will have higher-than-expected duty cycles. This is because of interactions with the `millis()` and `delay()` functions, which share the same internal timer used to generate those PWM outputs. This will be noticed mostly on low duty-cycle settings (e.g. 0 - 10) and may result in a value of 0 not fully turning off the output on pins 5 and 6.

PWM – 1 – Coding

```
int pwm = 9;  
byte pwmValue;           //8-bit, 1 byte, 0-255
```

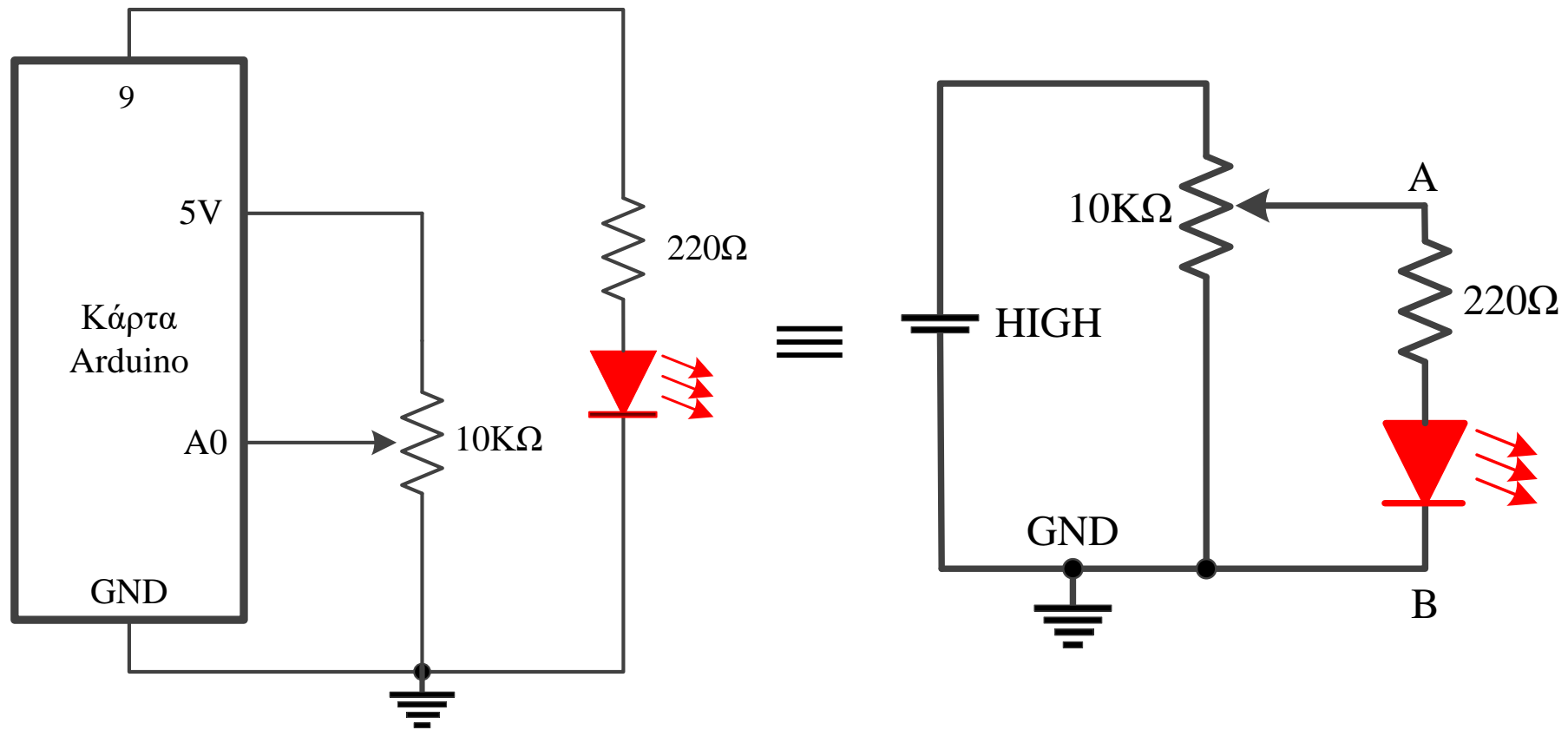
```
void setup()  
{  
}
```

```
void loop()  
{  
  analogWrite(pwm,pwmValue);  
  pwmValue++;  
  delay(20);  
}
```

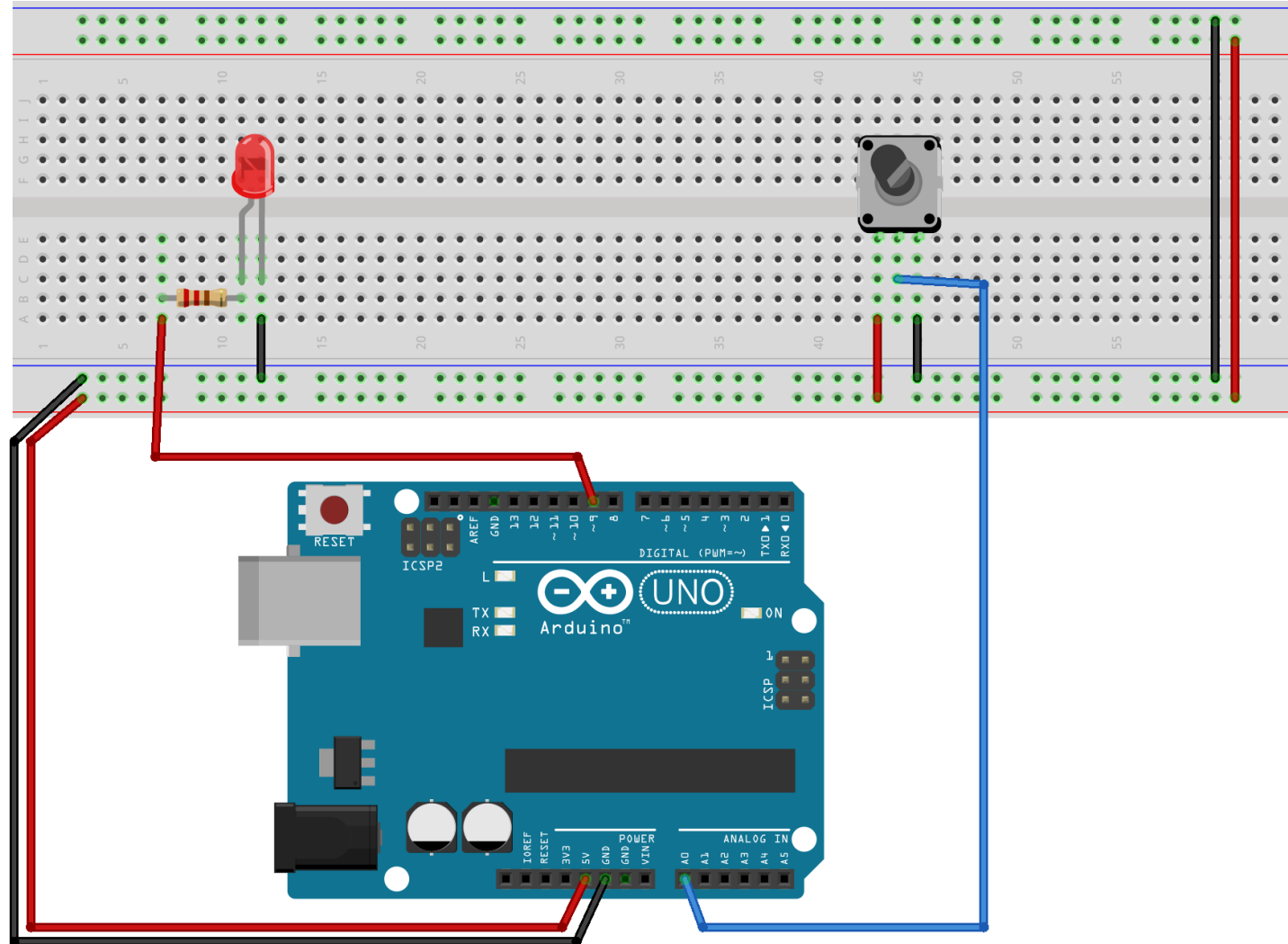
Εφαρμογές με PWM – 2

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα αυξομειώνει την φωτεινότητα ενός LED με τη χρήση ενός ποτενσιόμετρου και PWM.

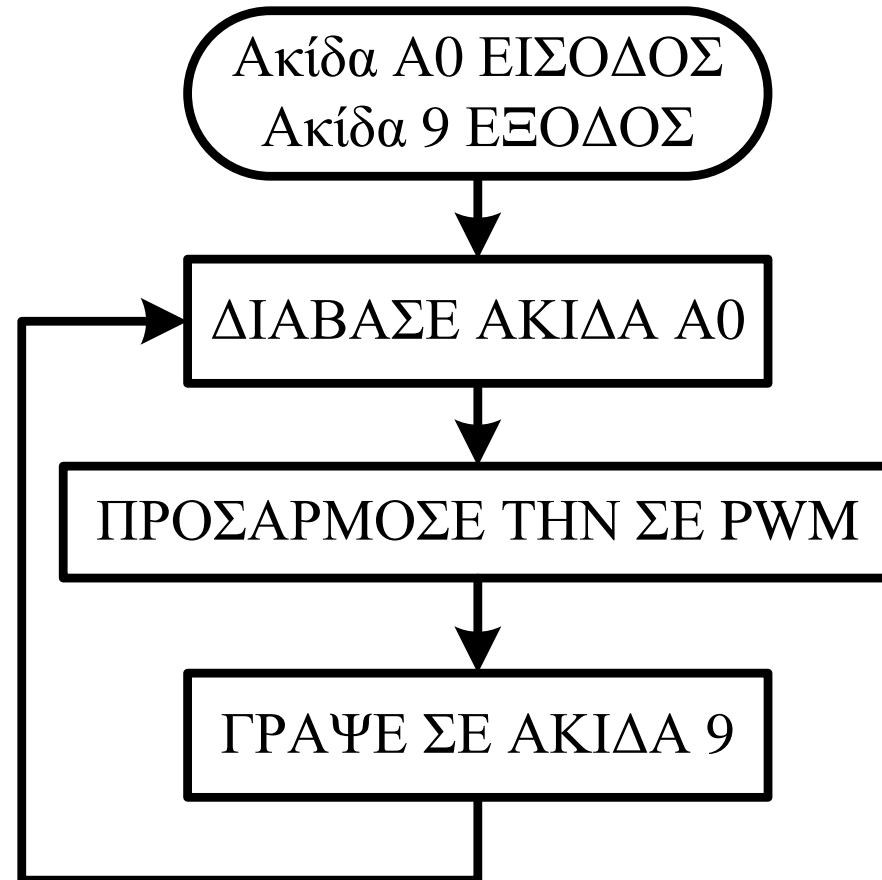
PWM – 2 – Το κύκλωμα



PWM – 2 – Το κύκλωμα



PWM – 2 – Λογικό Διάγραμμα



PWM – 2 – Coding

```
int analogValue = 0;           //Τιμή από την αναλογική είσοδο
int pwmValue = 0;             //Τιμή για το PWM
int analogPin = 0;           //Ονόμασε ακίδα A0 analogPin
int ledPin = 9;              //Ονόμασε ακίδα 9 ledPin

void setup()                  //Συνάρτηση αρχικοποίησης
{
  pinMode( ledPin, OUTPUT);   //Κάνε την ακίδα 9 έξοδο
}

void loop()                   //Κύριος βρόχος
{
  analogValue = analogRead(analogPin); //Ανάγνωση από την αναλογική
                                        //είσοδο (ποτενσιόμετρο)
  pwmValue = map(analogValue,0,1023,0,255); //Προσαρμογή κλίμακας
  analogWrite(ledPin,pwmValue);         //Οδήγηση LED
}
```

Εφαρμογές με PWM – 3

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino και Wiring C και θα αυξομειώνει την φωτεινότητα ενός LED δημιουργώντας fading:

A) από 0 έως 255

B) με καθορισμό της μέγιστης τιμής από την τιμή ενός ποτενσιόμετρου.

PWM – 3 – Coding

```
int analogValue = 0;
int pwmValue = 0;
int analogPin = 0;
int ledPin = 9;           // LED connected to digital pin 9

void setup() {
    // nothing happens in setup
}

void loop() {
    analogValue = analogRead(analogPin);
    pwmValue = map(analogValue,0,1023,0,255);
    // fade in from min to pwmValue in increments of 5 points:
    for (int fadeValue = 0 ; fadeValue <= pwmValue; fadeValue += 5) {
        // sets the value (range from 0 to pwmValue):
        analogWrite(ledPin, fadeValue);
        // wait for 30 milliseconds to see the dimming effect
        delay(30);}
}
```

Ανίχνευση εξωτερικών συμβάντων μέσω διακοπών (Ext. Interrupts)

Η τεχνική των διακοπών έχει εξαιρετική σημασία μιας και μέσω αυτής γίνεται ορθή διαχείριση εξωτερικών γεγονότων. Έστω λοιπόν ότι η εφαρμογή ελέγχει μια γραμμή παραγωγής. Για λόγους ασφαλείας, ο χρήστης θα πρέπει να έχει τη δυνατότητα να σταματήσει ακαριαία τη διαδικασία. Αυτό πρακτικά σημαίνει ότι το σύστημα, εκτός των άλλων, θα πρέπει να ανιχνεύει και την κατάσταση του κουμπιού. Χωρίς την τεχνική των διακοπών, θα υλοποιούνταν οι ακόλουθες λύσεις:

1. Συνεχής έλεγχος του κουμπιού. Αυτή η προσέγγιση έχει το μειονέκτημα ότι το σύστημα δε θα μπορούσε να εκτελέσει τίποτε άλλο παρά μόνο αυτό τον έλεγχο.
2. Έλεγχος του κουμπιού σε διάφορα σημεία του κώδικα. Ακόμα και αυτή η λύση θα λειτουργεί μόνο όταν το πάτημα του κουμπιού θα ταυτίζεται χρονικά με την εκτέλεση του κώδικα ελέγχου.

Επομένως, χρειάζεται μια εντελώς διαφορετική προσέγγιση. Με την τεχνική των διακοπών, όταν συμβεί κάποιο γεγονός (π.χ., ξαφνικό πάτημα κουμπιού), η ροή εκτέλεσης του προγράμματος διακόπτεται σε όποιο σημείο και αν βρίσκεται προκειμένου να ενεργοποιηθεί κάποια συνάρτηση που είναι υπεύθυνη για την εξυπηρέτηση του συμβάντος.

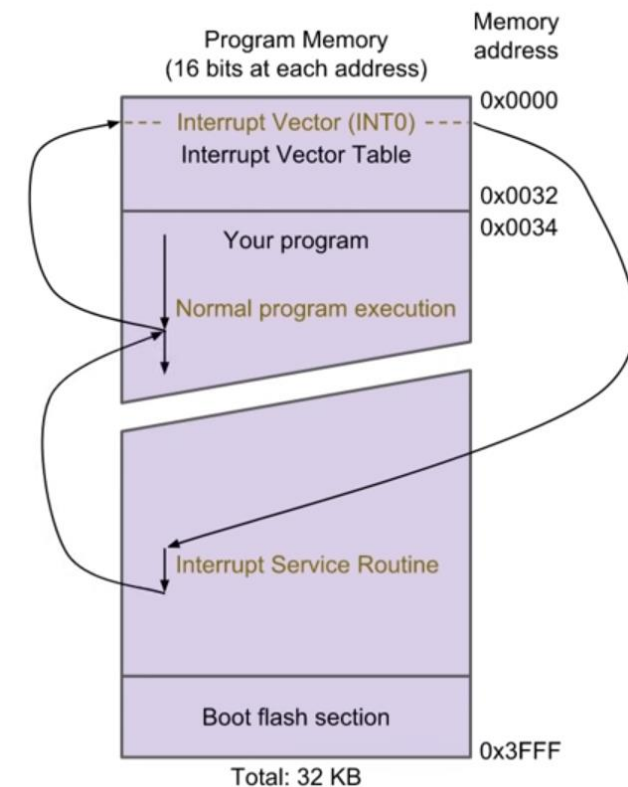
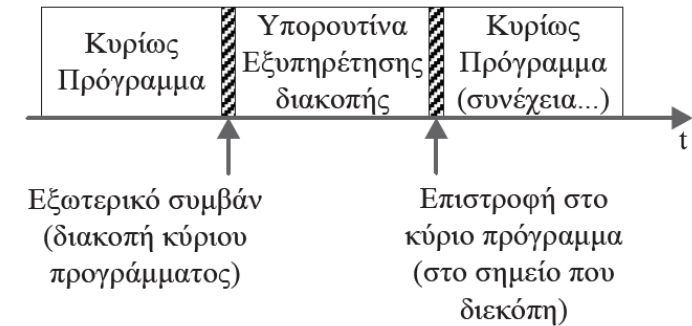
Ανίχνευση εξωτερικών συμβάντων μέσω διακοπών (Ext. Interrupts)

Ορισμένοι ψηφιακοί ακροδέκτες του Arduino υποστηρίζουν την τεχνική των διακοπών (**Interrupts**). Σε άλλες εκδόσεις υποστηρίζεται από λιγότερους ή και περισσότερους ακροδέκτες αυτή η τεχνική.

Η ανίχνευση συμβάντος γίνεται με τον έλεγχο της στάθμης σήματος σε έναν ή περισσότερους ακροδέκτες που υποστηρίζουν αυτή την τεχνική.

Η έκδοση UNO υποστηρίζει την τεχνική των διακοπών στους ακροδέκτες 2 και 3, οι οποίοι σε επίπεδο προγράμματος αντιστοιχούν στις διακοπές 0 και 1. Αυτό που θα πρέπει να κάνει ο προγραμματιστής είναι να αντιστοιχίσει τη συνάρτηση (υπορουτίνα) εξυπηρέτησης της διακοπής (ISR) η οποία θα εκτελείται κάθε φορά που θα ανιχνεύεται διακοπή σε έναν ακροδέκτη.

Το σχήμα δείχνει την εκτέλεση του κύριου προγράμματος και πώς αυτή διακόπτεται για να εκτελεστεί η συνάρτηση εξυπηρέτησης της διακοπής.



Ανίχνευση εξωτερικών συμβάντων μέσω διακοπών (Ext. Interrupts)

Η εντολή που θα πρέπει να τοποθετηθεί στη συνάρτηση `setup()` συντάσσεται ως εξής:

`attachInterrupt(int_no, function_name/ISR, mode)`

όπου:

`int_no` : ο αριθμός της διακοπής (0 ή 1) INT.0 (PIN2) ή INT.1(PIN3) για την έκδοση UNO.

`function_name/ISR` : το όνομα της συνάρτησης (υπορουτίνας) εξυπηρέτησης της διακοπής που θα καλείται όταν θα προκύψει το συμβάν.

`mode` : ο τρόπος ενεργοποίησης της διακοπής:

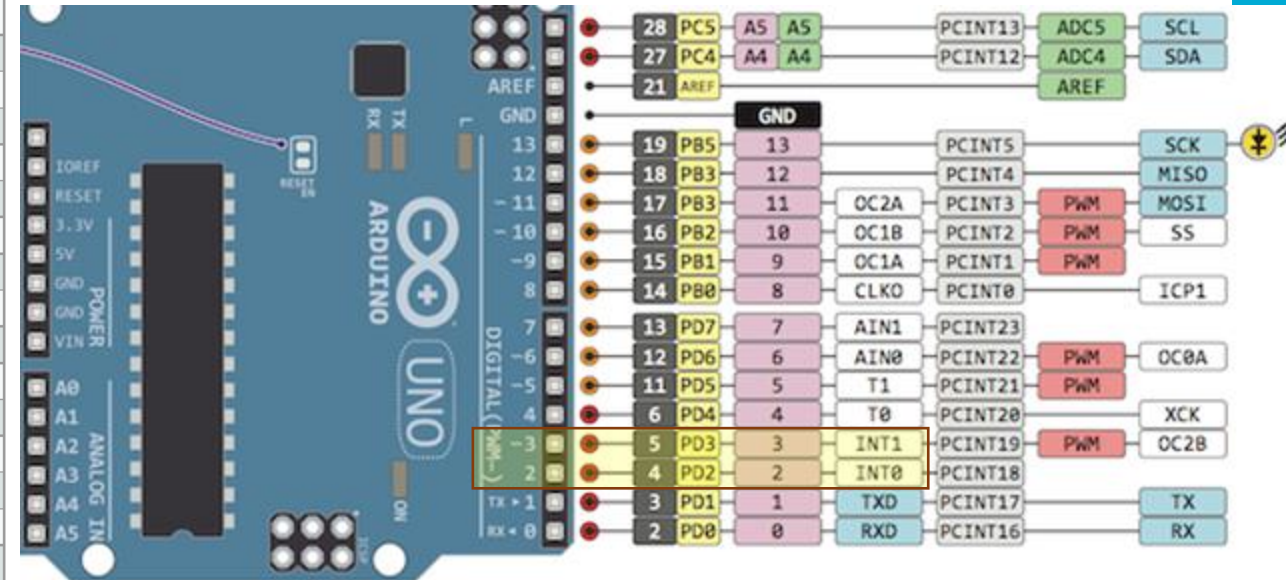
- **LOW** : όταν το σήμα είναι LOW,
- **CHANGE** : σε αλλαγή σήματος,
- **RISING** : σε μετάβαση σήματος από χαμηλή σε υψηλή στάθμη,
- **FALLING** : σε μετάβαση σήματος από υψηλή σε χαμηλή στάθμη.

Θα πρέπει εδώ να τονιστεί ξανά ότι το εξωτερικό συμβάν μπορεί να προκύψει οποιαδήποτε στιγμή, ενώ το κυρίως πρόγραμμα διακόπτεται (στον αμέσως επόμενο κύκλο ρολογιού) σε όποιο σημείο και να βρίσκεται η ροή εκτέλεσης.

External Interrupts

Table 11-1. Reset and Interrupt Vectors in ATmega328P

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x0002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 compare match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 compare match B
10	0x0012	TIMER2 OVF	Timer/Counter2 overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 capture event
12	0x0016	TIMER1 COMPA	Timer/Counter1 compare match A
13	0x0018	TIMER1 COMPB	Timer/Counter1 compare match B
14	0x001A	TIMER1 OVF	Timer/Counter1 overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 compare match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 compare match B
17	0x0020	TIMER0 OVF	Timer/Counter0 overflow
18	0x0022	SPI, STC	SPI serial transfer complete
19	0x0024	USART, RX	USART Rx complete
20	0x0026	USART, UDRE	USART, data register empty
21	0x0028	USART, TX	USART, Tx complete
22	0x002A	ADC	ADC conversion complete
23	0x002C	EE READY	EEPROM ready
24	0x002E	ANALOG COMP	Analog comparator
25	0x0030	TWI	2-wire serial interface
26	0x0032	SPM READY	Store program memory ready



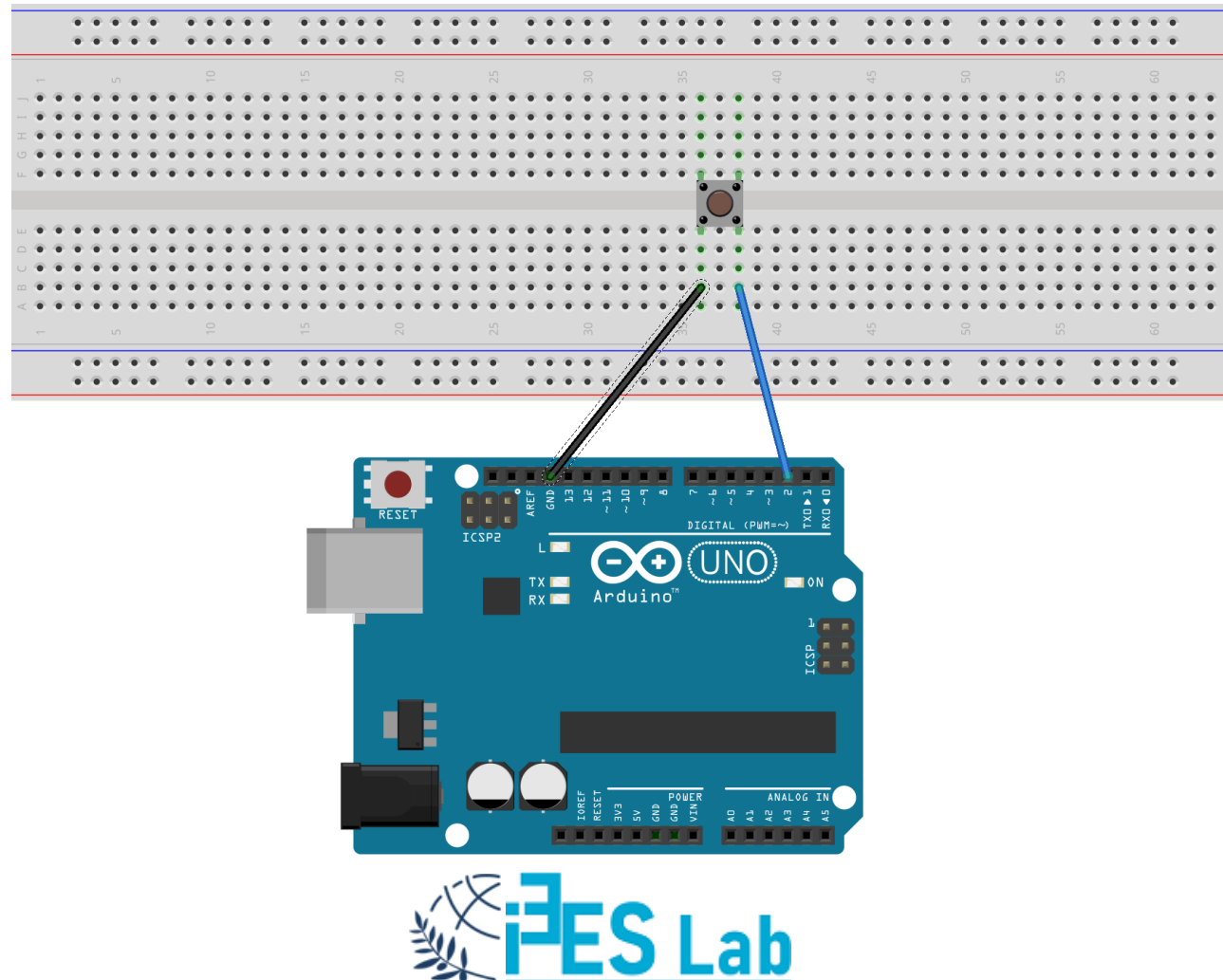
Ανίχνευση εξωτερικών συμβάντων μέσω διακοπών (Interrupts)

Όταν γράφουμε Interrupt Service Routines (ISR) προσέχουμε τα εξής:

1. Πάντα να είναι σύντομες διεργασίες / ρουτίνες.
2. Ποτέ να μην χρησιμοποιούμε καθυστερήσεις / delays.
3. Ποτέ να μην χρησιμοποιούμε Serial.Print
4. Πάντα οι μεταβλητές να ορίζονται ως volatile (οι μεταβλητές volatile αποθηκεύονται στη RAM και εξασφαλίζεται ότι είναι διαθέσιμες μεταξύ της ISR και του κυρίως προγράμματος).

Interrupts – 1

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino, Wiring C και Interrupts που θα αλλάζει την κατάσταση του LED στο PIN13 με την κλήση της διακοπής από ένα button στο INT.0 (PIN2).



Interrupts – 1

```
const byte ledPin = 13;  
const byte interruptPin = 2;  
volatile byte state = LOW; //state variable must be volatile with interrupts
```

```
void setup() {  
  pinMode(ledPin, OUTPUT);  
  pinMode(interruptPin, INPUT_PULLUP); // Pull-up ON  
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE); // PIN2=INT.0  
}
```

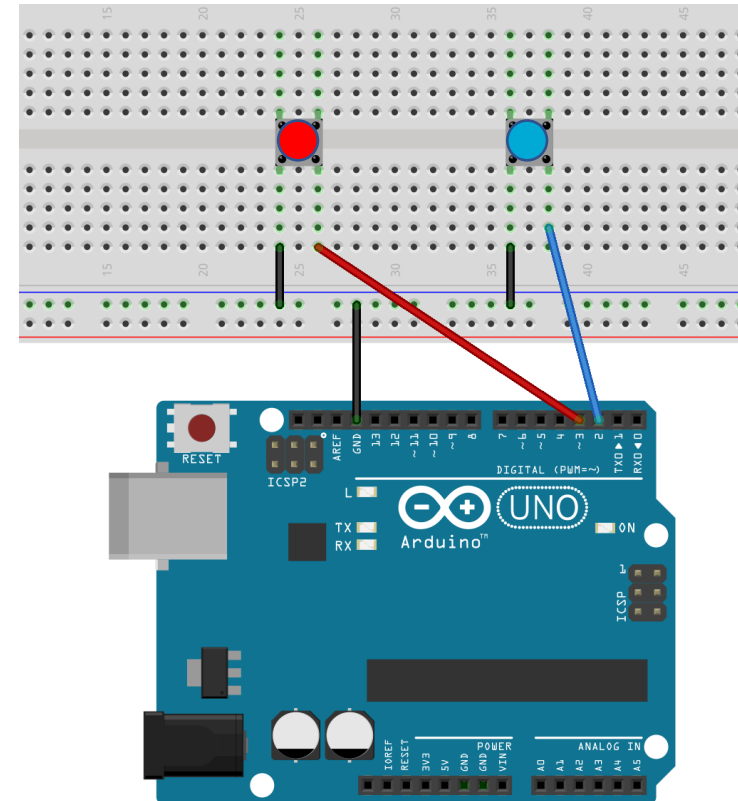
```
void loop() {  
  digitalWrite(ledPin, state);  
}
```

```
void blink() { //ISR function  
  state = !state;  
}
```

<https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/>

Interrupts – 2

Να δημιουργηθεί ένα κύκλωμα που θα χρησιμοποιεί Arduino, Wiring C και Interrupts που θα αλλάζει την κατάσταση του LED στο PIN13 με την κλήση διακοπών από δύο button INT.0 (PIN2) το μπλε (να το ανάβει) και INT.1 (PIN3) το κόκκινο (να το σβήνει). Η MCU πρέπει να βρίσκεται σε κατάσταση χαμηλής κατανάλωσης ισχύος.



Interrupts – 2

```
#include <LowPower.h> //Contributed library for Low Power
const byte ledPin = 13;
const byte interruptPin2 = 2;
const byte interruptPin3 = 3;
```

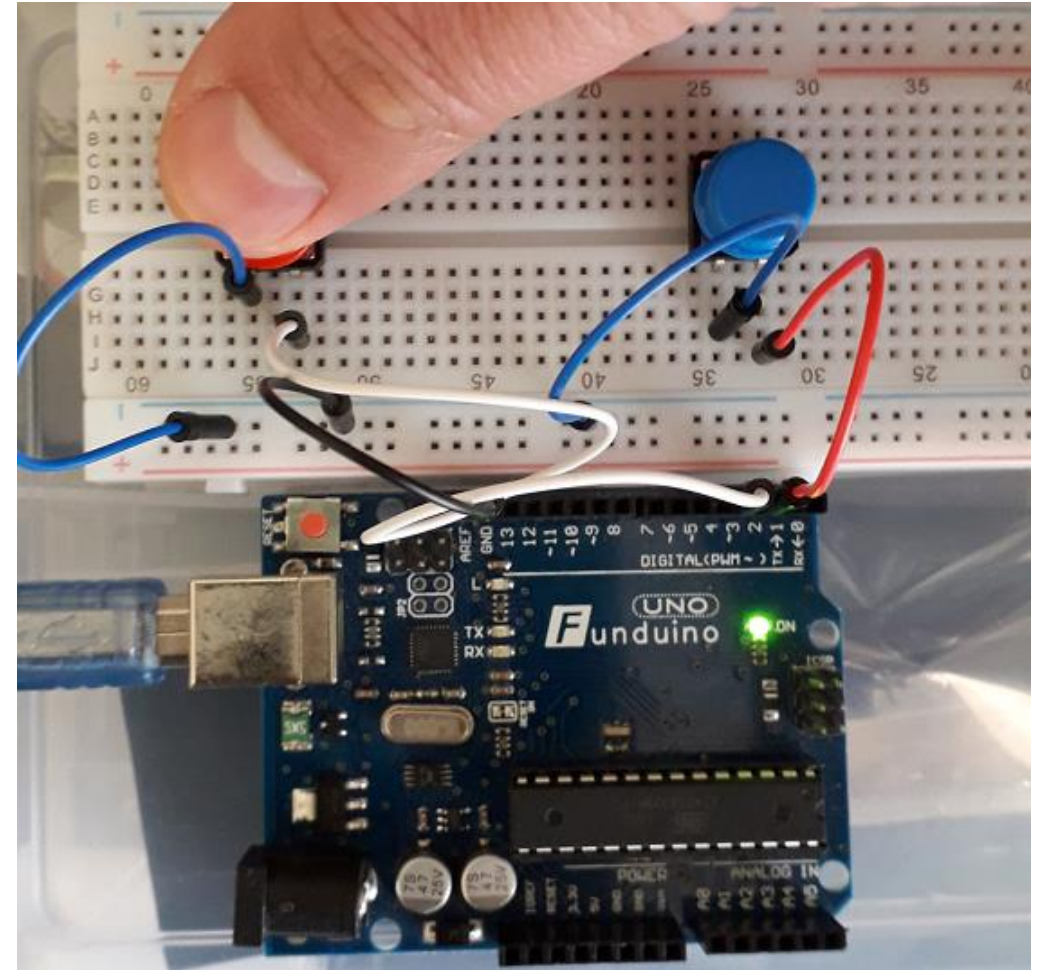
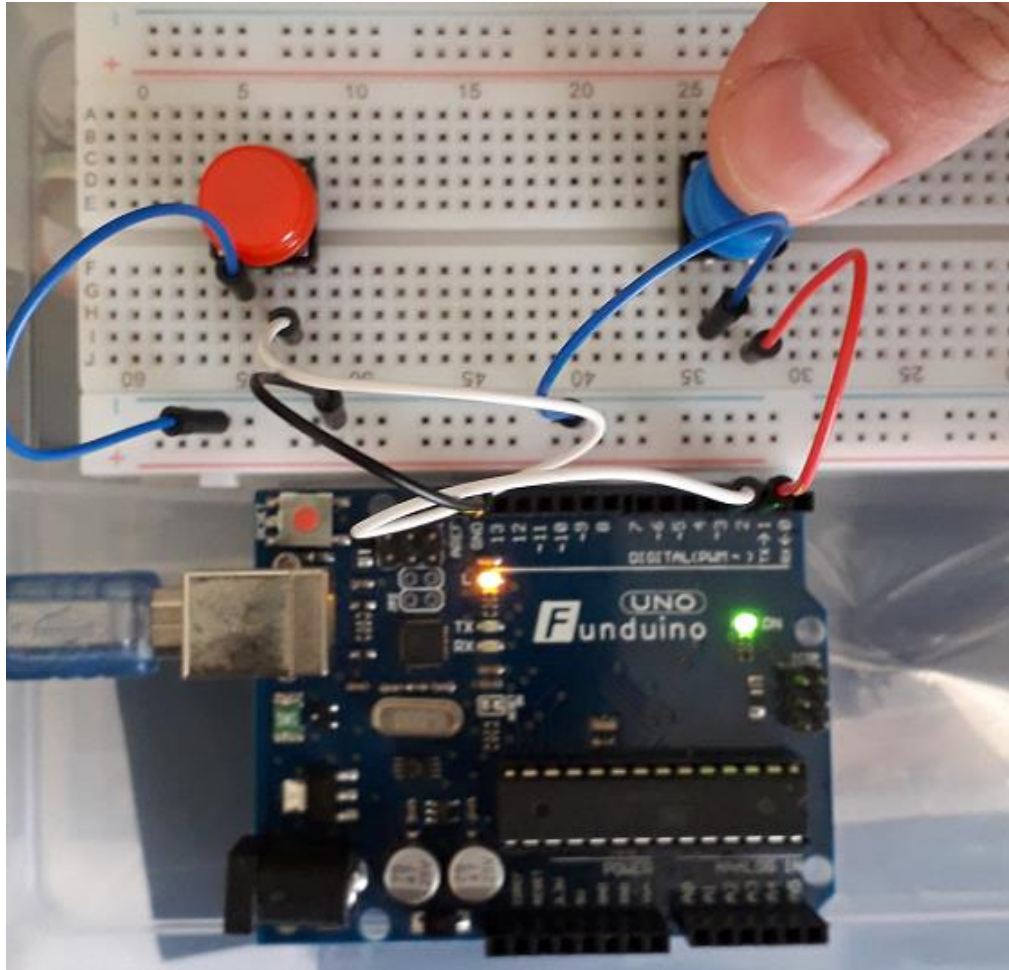
```
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin2, INPUT_PULLUP);
  pinMode(interruptPin3, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin2), turnLEDOn, CHANGE);
  attachInterrupt(digitalPinToInterrupt(interruptPin3), turnLEDOff, CHANGE);}
```

```
void loop() {
  LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF); } // MCU sleeps until Interrupt comes
```

```
void turnLEDOn() {
  digitalWrite(ledPin,HIGH);}
```

```
void turnLEDOff() {
  digitalWrite(ledPin,LOW);}
```

Interrupts – 2



Contributed Libraries for Arduino

1. Συνήθως είναι σε .zip αρχεία και πρέπει να τις προσθέσουμε στο IDE.

Sketch>Include Library>Add .ZIP Library

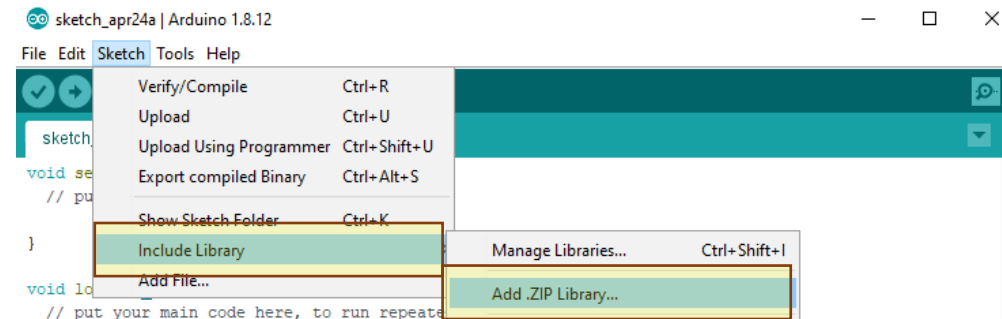
2. Η βιβλιοθήκη βρίσκεται σε κάποιο github ή την έχουμε δημιουργήσει εμείς και την επιλέγουμε από εκεί που είναι αποθηκευμένη στον δίσκο μας. Αν δεν υπάρχει κάποιο πρόβλημα η βιβλιοθήκη εγκαθίσταται κανονικά.

3. Τέλος, για να την εισάγουμε στο sketch θα πρέπει να την επιλέξουμε από τις διαθέσιμες βιβλιοθήκες στην λίστα.

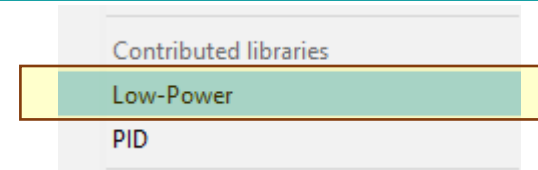
Sketch>Include Library>Low-Power

Και θα δούμε το .h αρχείο στον κώδικά μας.

4. Τώρα μπορούμε να καλούμε τις μεθόδους μέσα στον κώδικα και σύμφωνα με τα lib specs.



Library added to your libraries. Check "Include library" menu



asm – inline Assembly – Arduino IDE

- Ένα **inline assembly statement** είναι μια συμβολοσειρά από χαρακτήρες (string) που ορίζουν εντολές/κώδικα assembly.
- Αυτή η συμβολοσειρά μπορεί να περιέχει όσες εντολές (instructions) αναγνωρίζονται από τον assembler και ψευδοεντολές (directives).
- Ο GCC compiler δεν αναλύει (parse) τις εντολές του assembler και δεν γνωρίζει τη λειτουργία τους και αν είναι σωστές.
- Πολλές εντολές assembly μπορούν να τοποθετηθούν σε μια μόνο **asm** δήλωση.
- Κάθε inline asm δήλωση περιβάλλεται από παρενθέσεις και ακολουθεί το compiler keyword: **asm** ή **__asm__**
- Κάθε εντολή assembly περιβάλλεται από διπλά εισαγωγικά “code \n” και τερματίζει με τον χαρακτήρα αλλαγής γραμμής, “\n” ή “\n\t” για την μορφοποίηση του κώδικα. Για παράδειγμα:

asm(“code \n”);



```
sketch_asm_blinking_LED13 | Arduino 1.8.12
File Edit Sketch Tools Help
sketch_asm_blinking_LED13 $

void setup() {
  asm("sbi 0x04,5");
}

void loop() {
  asm("start:");
  asm("clr r16 \n\t clr r17");
  asm("sbi 0x05,5");
  asm("ldi r18, 5");
  asm("rcall delay");
  asm("cbi 0x05,5");
  asm("ldi r18, 5");
  asm("rcall delay");
  asm("rjmp start");

  asm("delay:");
  asm("dec r16");
  asm("brne delay");
  asm("dec r17");
  asm("brne delay");
  asm("dec r18");
  asm("brne delay");
  asm("ret");
}

Done uploading.
Sketch uses 478 bytes (1% of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0% of dynamic memory, leaving 2039 bytes for local variables. Maximum is
Arduino Uno on COM4
```

asm – inline Assembly – Blinking LED13

```
void setup() {  
  asm("sbi 0x04,5");  
  asm("rjmp start");  
  void loop() {  
    asm("start:");  
    asm("clr r16 \n\t clr r17");  
    asm("sbi 0x05,5");  
    asm("ldi r18, 5");  
    asm("rcall delay");  
    asm("cbi 0x05,5");  
    asm("ldi r18, 5");  
    asm("rcall delay");  
    asm("rjmp start");  
    asm("delay:");  
    asm("dec r16");  
    asm("brne delay");  
    asm("dec r17");  
    asm("brne delay");  
    asm("dec r18");  
    asm("brne delay");  
    asm("ret");  
  }  
}
```

// δήλωση του PIN13 ως έξοδο, DDRB5=1
//μετάβαση στην θέση .org του code segment

//αρχή του code segment
//μηδενισμός καταχωρητών r16 και r17
//ορίζει PORTB5=1
//φορτώνει Imm 5 στον r18
//κλήση delay
//ορίζει PORTB5=0
//φορτώνει Imm 5 στον r18
//κλήση delay
//επανεκκίνηση του κώδικα από το start
//αρχή ρουτίνας καθυστέρησης delay
//μειώνει κατά ένα τον r16
//κάνει διακλάδωσή αν Z=0
//μειώνει κατά ένα τον r17
//κάνει διακλάδωσή αν Z=0
//μειώνει κατά ένα τον r18
//κάνει διακλάδωσή στο delay μέχρι ο r18 να μηδενίσει
// επιστρέφει από την υπορουτίνα στο κυρίως μέρος του προγράμματος.

asm – inline Assembly – Πρόσθεση δύο αριθμών

```
volatile byte a=0;           //δήλωση της μεταβλητής a και αποθήκευσή της στην SRAM
void setup() {
  Serial.begin(9600);        //ενεργοποίηση της σειριακής
  asm("rjmp start");         //μετάβαση στην θέση .org του code segment
}
void loop() {
  asm("start:");             //αρχή του code segment
  asm("ldi r16, 0x30");       //φόρτωση της τιμής $30 στον καταχωρητή r16
  asm("inc r16");             //αύξηση κατά ένα της τιμής του r16
  asm("ldi r17, 0x1");        //φόρτωση της τιμής $1 στον καταχωρητή r17
  asm("add r17, r16");        //πρόσθεση του r17 με τον r16 και αποθήκευση στον r17
  asm("sts (a), r17");        //απευθείας αποθήκευση των δεδομένων του r17 στη SRAM θέση μνήμης a
  Serial.print(a, BIN);       //εκτύπωση serial monitor της a σε bin, dec και hex
  Serial.print(" - ");
  Serial.print(a, DEC);
  Serial.print(" - ");
  Serial.println(a, HEX);
}
```

Χρήσιμοι Σύνδεσμοι

Το λογισμικό Arduino IDE είναι διαθέσιμο στον ιστοχώρο:

<https://www.arduino.cc/en/Main/Software>

Η γλώσσα προγραμματισμού και οι εντολές που υποστηρίζει είναι διαθέσιμες στο:

<https://www.arduino.cc/reference/en/>

Για την υλοποίηση σχεδίων είναι διαθέσιμο το ανοιχτό λογισμικό:

<https://www.electroschematics.com/fritzing-software-download/>

Αναφορές – References

- “Οργάνωση και Σχεδίαση Υπολογιστών”, Πογαρίδης Δημήτριος, ΔΙΣΙΓΜΑ, 2019.
- “Ενσωματωμένα συστήματα: Ο αθέατος ψηφιακός κόσμος”, Δασυγένης Μηνάς, Σούντρης Δημήτριος, ΚΑΛΛΙΠΟΣ.
- “Microprocessor Theory and Applications with 68000/68020 and Pentium”, M. RAFIQUZZAMAN, WILEY, 2008.
- **SEE 3223: Microprocessors** - Department of Control and Mechatronic Engineering (CMED), Faculty of Electrical Engineering, Universiti Teknologi Malaysia.
- CSULB – EE346 – Course “Microprocessor Principles and Applications”.
- Motorola Microcontrollers.
- Microchip/Atmel AVR.
- Sparkfun.