

Μάθετε το Arduino



Μέρος πρώτο

Εισαγωγή

Χατζηκυριάκου Γιώργος - 2009

Περιεχόμενα

Τι είναι το Arduino;	3
Το περιβάλλον ανάπτυξης	4
Ρυθμίσεις του περιβάλλοντος ανάπτυξης	5
Δομή προγράμματος	5
Μεταβλητές	5
Σταθερές	6
Πίνακες – Arrays	6
Αριθμητικοί τελεστές	6
Τελεστές σύγκρισης	7
Λογικοί τελεστές	7
Ψηφιακή Έξοδος	8
Ψηφιακή Είσοδος	10
Ασκήσεις	12

Τι είναι το Arduino;

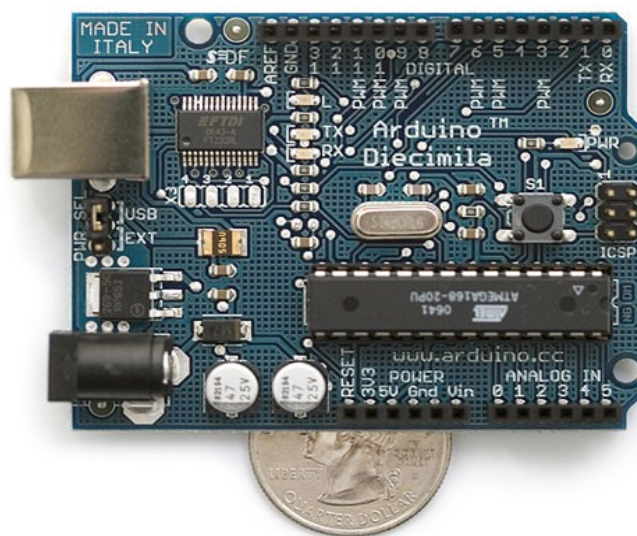
Το Arduino είναι ένας μικροελεγκτής ο οποίος περιλαμβάνει ένα chip ATmega. Με λίγα λόγια διαθέτει εισόδους και εξόδους που αντιδρούν βάση του προγραμματισμού που κάναμε και που φορτώσαμε στο chip με τη βοήθεια του υπολογιστή. Η γλώσσα προγραμματισμού που χρησιμοποιεί είναι η Wiring, η οποία είναι αρκετά εύκολη στη σύνταξη και διατίθεται σε πλατφόρμες Linux, MAC και Windows με άδεια χρήσης GPL.

Αυτό όμως που κάνει το Arduino ακόμα πιο σημαντικό είναι ότι όλο το κύκλωμα της πλακέτας διατίθεται με άδεια χρήσης Creative Commons, πράγμα που σημαίνει ότι ο καθένας μπορεί να κατασκευάσει την δική του πλακέτα όπως αυτός θέλει. Φυσικά για τους αρχάριους το να κατασκευάσουν την πλακέτα μόνοι τους, ίσως ακουστεί λίγο τραβηγμένο, οπότε είναι ευκολότερο να αγοράσουν μια έτοιμη πλακέτα Arduino από το διαδίκτυο η οποία διατίθεται σε πάρα πολύ προσιτή τιμή.

Αν και μικροσκοπικό (7x5 cm) οι δυνατότητες που προσφέρει είναι πάρα πολλές. Μπορούμε να το χρησιμοποιήσουμε σε εφαρμογές ρομποτικής και γενικότερα σε αυτοματισμούς καταφέροντας έτσι πάρα πολλά όπως: την κίνηση servo, stepper και DC κινητήρων, τη λήψη πληροφοριών από διάφορους αισθητήρες (θερμοκρασίας, υγρασίας, υπέρυθρων κ.α), την αμφίδρομη σειριακή επικοινωνία μεταξύ Arduino και PC χρησιμοποιώντας γλώσσες προγραμματισμού (όπως Java και python), όπως επίσης την αναπαραγωγή και αντίληψη ήχων.

Φυσικά οι δυνατότητες του Arduino δεν σταματούν εκεί, στο site του Arduino (<http://arduino.cc/>) θα ανακαλύψετε μια μεγάλη κοινότητα με αρκετές πληροφορίες όσο αφορά τις εκδόσεις την αγορά και το προγραμματισμό της πλακέτας.

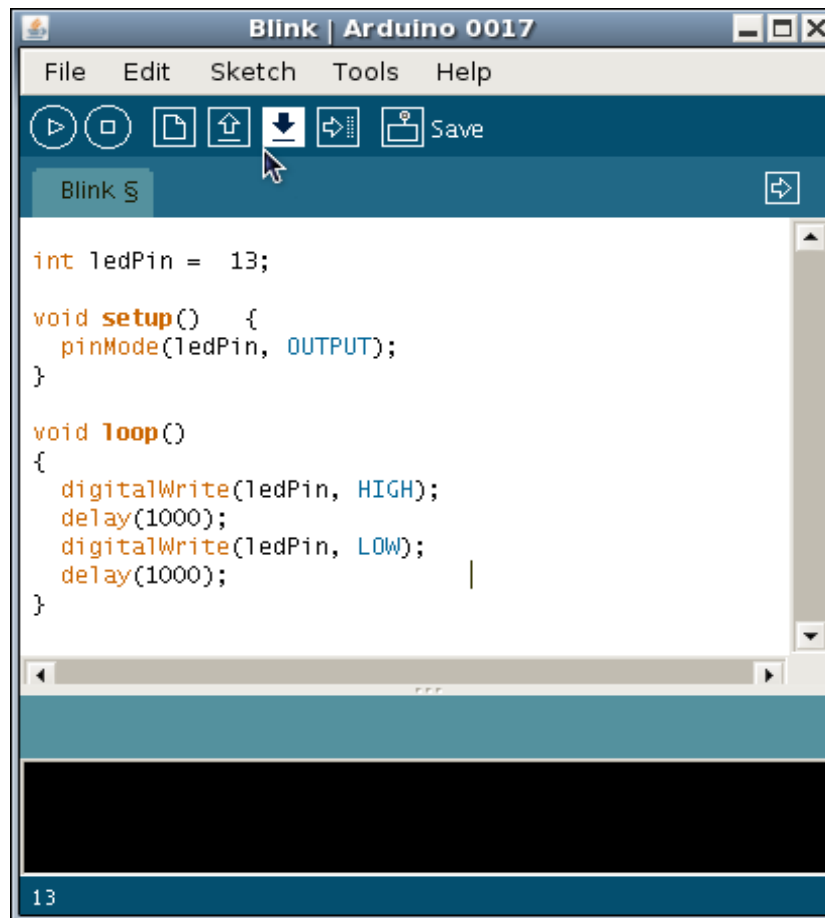
Η πλακέτα Arduino μέχρι αυτή τη στιγμή διατίθεται σε 12 βασικές παραλλαγές οι οποίες αναφέρονται σε διαφορετικές χρήσεις η κάθε μια, ανάλογα με τις ανάγκες της εφαρμογής μας.



Εικόνα 1: Το Arduino Diecimila








Το περιβάλλον ανάπτυξης

Το περιβάλλον ανάπτυξης (IDE) του Arduino είναι μία πολυπλατφορμική εφαρμογή γραμμένη σε Java και βασίζεται στο περιβάλλον της γλώσσας προγραμματισμού Processing (<http://processing.org/>).



Εικόνα 2: Το περιβάλλον ανάπτυξης του Arduino

Βασικές λειτουργίες του IDE:

	Έλεγχος του κώδικα για λάθη.
	Τερματισμός της σειριακής κονσόλας.
	Δημιουργία νέου έργου (sketch).
	Παρουσίαση μενού με όλα τα αποθηκευμένα έργα. Πατώντας σε ένα από αυτά ανοίγει για επεξεργασία.
	Αποθήκευση του έργου.
	Μεταγλώττιση του κώδικα και ανέβασμα του στο Arduino.
	Εμφάνιση της σειριακής κονσόλας. Αποστολή και λήψη δεδομένων που στάλθηκαν μέσω της σειριακής θύρας,

Ρυθμίσεις του περιβάλλοντος ανάπτυξης

Η βασικές ρυθμίσεις που πρέπει να κάνουμε από την στιγμή που ενώσουμε το Arduino στο σύστημα μας είναι:

1. Επιλογή πλακέτας. Από το μενού Tools -> Board επιλέγουμε την πλακέτα που έχουμε. Στο συγκεκριμένο οδηγό θα χρησιμοποιήσουμε το Arduino Diecimila, οπότε επιλέγουμε το “Arduino Diecimila, Duemilanove or Nano w/ Atmega168”.
2. Επιλογή σειριακής θύρας. Από το μενού Tools -> Serial Port επιλέγουμε την σειριακή θύρα ή θύρα USB που έχουμε συνδεδεμένο το Arduino (πχ. /dev/ttyUSB0 σε ΛΣ Linux).

Ρυθμίσεις που αφορούν το μέγεθος του κειμένου, τον φάκελλο αποθήκευσης, χρήση εξωτερικού κειμενογράφου βρίσκονται στη καρτέλα Preferences (File -> Preferences). Για περισσότερες ρυθμίσεις μπορούμε να κάνουμε αλλαγές το αρχείο preferences.txt (βρίσκεται στον φάκελο του χρήστη ~/.arduino/preferences.txt στο ΛΣ Linux).

Δομή προγράμματος

Ένα τυπικό πρόγραμμα του Arduino έχει την εξής δομή:

```
// δηλώσεις μεταβλητών  
  
void setup() {  
    // αρχικοποιήσεις  
}  
  
void loop() {  
    // ...  
}
```

Όπως βλέπουμε υπάρχουν δυο βασικές συναρτήσεις σε ένα τυπικό πρόγραμμα.

Η συνάρτηση setup() εκτελείται στην αρχή του προγράμματος και για μία μόνο φορά. Χρησιμοποιείται για τις αρχικοποιήσεις των μεταβλητών, τις δηλώσεις των pin (αν θα είναι είσοδος ή έξοδος) και τις αρχικοποιήσεις των βιβλιοθηκών.

Η συνάρτηση loop() κάνει αυτό που λέει και το όνομά της, Ο κώδικας που γράφεται μέσα στη συνάρτηση αυτή επαναλαμβάνεται συνεχώς δίνοντας την δυνατότητα στο πρόγραμμα μας να αλλάζει τιμές και το Arduino να ανταποκρίνεται ανάλογα.

Μεταβλητές

Μεταβλητή στις γλώσσες προγραμματισμού γενικά ονομάζουμε ένα γλωσσικό αντικείμενο που μπορεί να λάβει διάφορες τιμές, μία κάθε φορά. Οι τιμές μιας μεταβλητής περιορίζονται συνήθως σε ένα τύπο δεδομένων.

Οι βασικοί τύποι δεδομένων στο Arduino είναι:

1. `byte`: αποθηκεύει μια αριθμητική τιμή 8-bit χωρίς δεκαδικά ψηφία, παίρνουν τιμές από 0 μέχρι 255.
2. `int`: ακραίοι, παίρνουν τιμές από -32,768 μέχρι 32767.
3. `long`: μεγάλοι μεγέθους ακαριαίοι, παίρνουν τιμές από -2,147,483,648 μέχρι 2,147,483,647
4. `float`: πραγματικοί αριθμοί, παίρνουν τιμές από 3.4×10^{-38} μέχρι 3.4×10^{38}

Τις μεταβλητές μπορούμε να τις δηλώσουμε στην αρχή του προγράμματός μας:

```
int myvariable;
```

Μπορούμε επίσης να δώσουμε αρχική τιμή στη μεταβλητή ταυτόχρονα με τη δήλωσή της:

```
int myvariable = 47;
```

Σταθερές

Σταθερές είναι αντικείμενα τα οποία παίρνουν μόνο μία τιμή, και δηλώνονται μαζί με τις μεταβλητές:

```
#define ledPin 13
```

Πίνακες – Arrays

Πίνακα ονομάζουμε διάταξη δεδομένων μιας ή περισσότερων διαστάσεων η οποία είναι συγκεκριμένου τύπου δεδομένων. Για παράδειγμα αν έχουμε ένα πίνακα ακραίων 5 θέσεων τον οποίο ονομάζουμε `myarray` τον δηλώνουμε όπως βλέπουμε παρακάτω:

```
int myarray[5];
```

Για να δώσουμε τιμή στο τέταρτο στοιχείο του πίνακα `myarray` γραφτούμε:

```
myarray[3] = 12;
```

επίσης μπορούμε να γεμίσουμε τον πίνακα ταυτόχρονα με την δήλωση του:

```
int myarray[] = {12, 45, 32, 61, 55};
```

Αριθμητικοί τελεστές

Οι αριθμητικοί τελεστές καλύπτουν τις βασικές πράξεις: πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση (+, -, *, /). Για παράδειγμα μπορούμε να κάνουμε την πρόσθεση δύο ακέραιων και το αποτέλεσμα να εκχωρηθεί σε μία μεταβλητή:

```
sum = 458 + 954;
```

Τελεστές σύγκρισης

Με τους τελεστές σύγκρισης μπορούμε να ελέγξουμε αν μία συγκεκριμένη συνθήκη μεταξύ μεταβλητών ή σταθερών είναι “Αληθής”. Ποιοι συγκεκριμένα υπάρχουν οι παρακάτω τελεστές σύγκρισης στο Arduino:

$x == y$	το x είναι ίσο με το y
$x != y$	το x είναι άνισο του y
$x < y$	το x είναι μικρότερο με το y
$x > y$	το x είναι μεγαλύτερο με το y
$x <= y$	το x είναι μικρότερο ή ίσο με το y
$x >= y$	το x είναι μεγαλύτερο ή ίσο με το y

Λογικοί τελεστές

Με τους λογικούς τελεστές μπορούμε να συγκρίνουμε δύο ή περισσότερες εκφράσεις, δίνοντας αποτέλεσμα “Αληθής” ή “Ψευδής”. Υπάρχουν τρεις λογικοί τελεστές:

Λογικό ΚΑΙ	&& - επιστρέφει “Αληθής” αν όλες οι εκφράσεις είναι “Αληθείς”
Λογικό Ή	- επιστρέφει “Αληθής” αν μία από τις εκφράσεις είναι “Αληθείς”
Λογικό ΟΧΙ	! - επιστρέφει “Αληθής” αν η έκφραση είναι “Ψευδής”

Παράδειγμα:

```
if(x > 0 && x < 5){  
    //κώδικας  
}
```

Στο παραπάνω κομμάτι κώδικα γίνεται έλεγχος αν το x είναι μεγαλύτερο από το 0 **ΚΑΙ** μικρότερο από 5 τότε εκτελείται ο κώδικας που βρίσκεται μέσα στις αγκύλες. Με λίγα λόγια η πρόταση *If()* ελέγχει **αν** η συνθήκη μέσα στις παρενθέσεις είναι “Αληθής”.

Ένα άλλο παράδειγμα:

```
if(!x > 0){  
    //κώδικας  
}
```

Εδώ γίνεται έλεγχος αν το x είναι μεγαλύτερο από 0, αν αυτή η συνθήκη **ΔΕΝ** ισχύει τότε έχουμε το αποτέλεσμα “Αληθής” και εκτελείται ο κώδικας μέσα στις αγκύλες.

Ψηφιακή Έξοδος

Το Arduino Diecimila αποτελείται από δεκατρία ψηφιακά pin, τα οποία μπορούμε να τα χρησιμοποιήσουμε το κάθε ένα ξεχωριστά, είτε για είσοδο είτε για έξοδο. Μπορούμε να τα προγραμματίσουμε να συμπεριφέρονται όπως εμείς θέλουμε, αρκεί να κάνουμε τις σωστές δηλώσεις στο κώδικα που θα φορτώσουμε στη πλακέτα.

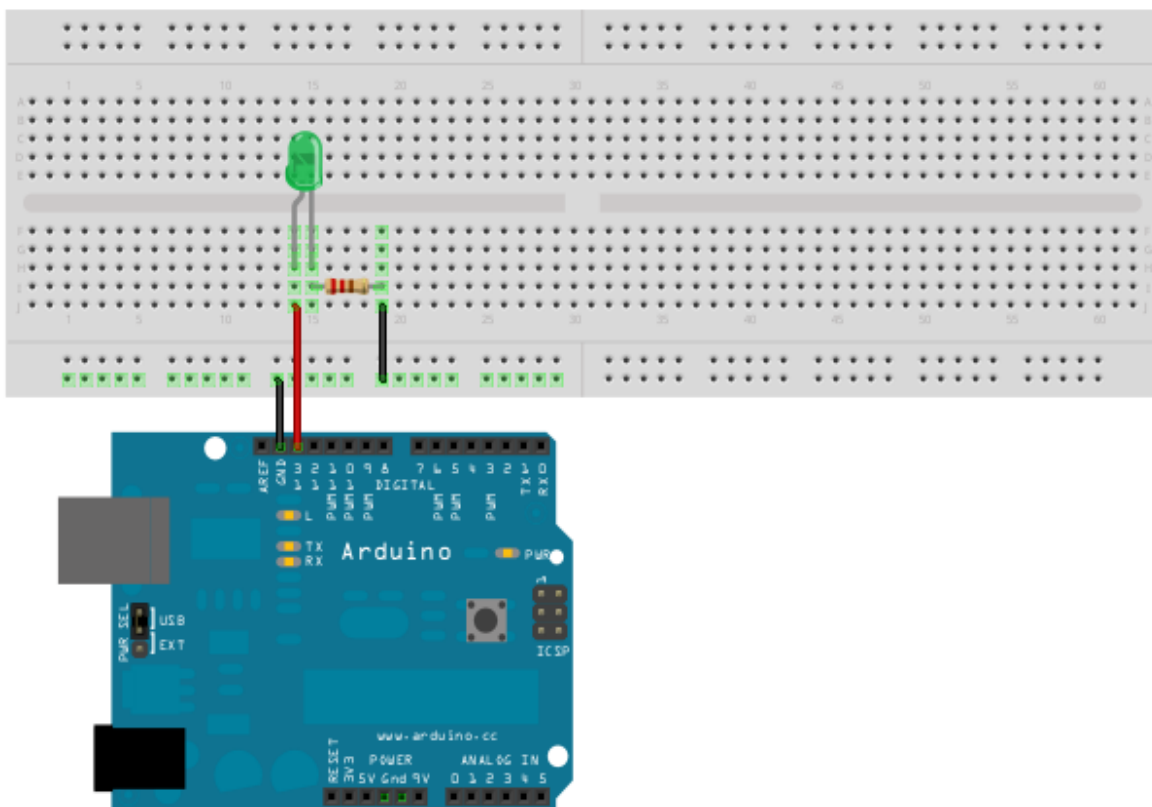
Η έξοδος του κάθε pin μπορεί να προγραμματιστεί να δίνει τιμές HIGH ή LOW. Λέγοντας HIGH ενώνουμε το δυαδικό '1' και έχουμε τάση εξόδου 5V DC, ενώ το LOW είναι το δυαδικό '0' και έχει τάση εξόδου 0V DC (ground).

Παρακάτω θα δούμε ένα παράδειγμα για να μπορέσουμε να κατανοήσουμε τον προγραμματισμό των ψηφιακών pin του Arduino. Θα ενώσουμε ένα led στη board και θα το προγραμματίσουμε να ανάβει και να σβήνει σε χρονικά διαστήματα ενός δευτερολέπτου.

Τα υλικά που θα χρειαστούμε στη πρώτη εφαρμογή είναι:

- 1 x breadboard
- 1 x led
- 1 x αντίσταση 220Ω (κόκκινο – κόκκινο - καφέ)

Η συνδεσμολογία που κάναμε φαίνεται στο παρακάτω σχήμα:



Εικόνα 3: Φυσική αναπαράσταση του κυκλώματος

Για την φυσική αναπαράσταση του κυκλώματος έγινε χρήση του προγράμματος Fritzing (<http://fritzing.org/>).

Ας μελετήσουμε τώρα το πρόγραμμα που θα φορτώσουμε στο Arduino:

```
int ledPin = 13;

void setup(){
  pinMode(ledPin, OUTPUT);
}

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



Στη πρώτη γραμμή δηλώνουμε μια ακέραια μεταβλητή την *ledPin* και της δίνουμε την αρχική τιμή 13, ο λόγος που έγινε αυτή η δήλωση είναι για να μας διευκολύνει αν δεν θυμόμαστε σε ποιο pin συνδέσαμε το led.

Στη συνάρτηση *setup()* η εντολή *pinMode()* χρησιμοποιείται για να δηλώσουμε την χρήση του pin 13, δηλαδή αν θα το χρησιμοποιήσουμε σαν είσοδο (input) ή σαν έξοδο (output).

Ας πάμε τώρα στη συνάρτηση *loop()* που όπως είπαμε ποιο πριν χρησιμοποιείται για την επανάληψη του κώδικα, δίνοντάς μας τη ευκαιρία αν θέλουμε να ανανεώσουμε τις τιμές των μεταβλητών και να κάνουμε το Arduino να ανταποκρίνεται στις αλλαγές αυτές.

Εδώ συναντάμε την εντολή *digitalWrite()* που είναι υπεύθυνη στο να δώσει τιμές HIGH ή LOW στο pin που δηλώνουμε μέσα στις παρενθέσεις.

Η εντολή *delay()* είναι υπεύθυνη για την παύση του προγράμματός μας για χρόνο που δηλώνουμε μέσα στις παρενθέσεις, το χρόνο αυτό τον καθορίζουμε σε microseconds (ms) (1000ms = 1sec).

Αφού ελέγξουμε τον κώδικά για τυχόν λάθη πατώντας το κουμπί  , πατάμε το κουμπί  για να γίνει η μεταγλώττιση και το ανέβασμα του δεκαεξαδικού αρχείου στο μικροελεγκτή. Αν όλα πήγαν καλά τότε θα δούμε το led να αναβοσβήνει σε διαστήματα ενός δευτερολέπτου.

Ψηφιακή Είσοδος

Στο προηγούμενο παράδειγμα είδαμε πώς μπορούμε να δηλώσουμε ένα pin του Arduino και να το χρησιμοποιήσουμε σαν έξοδο.

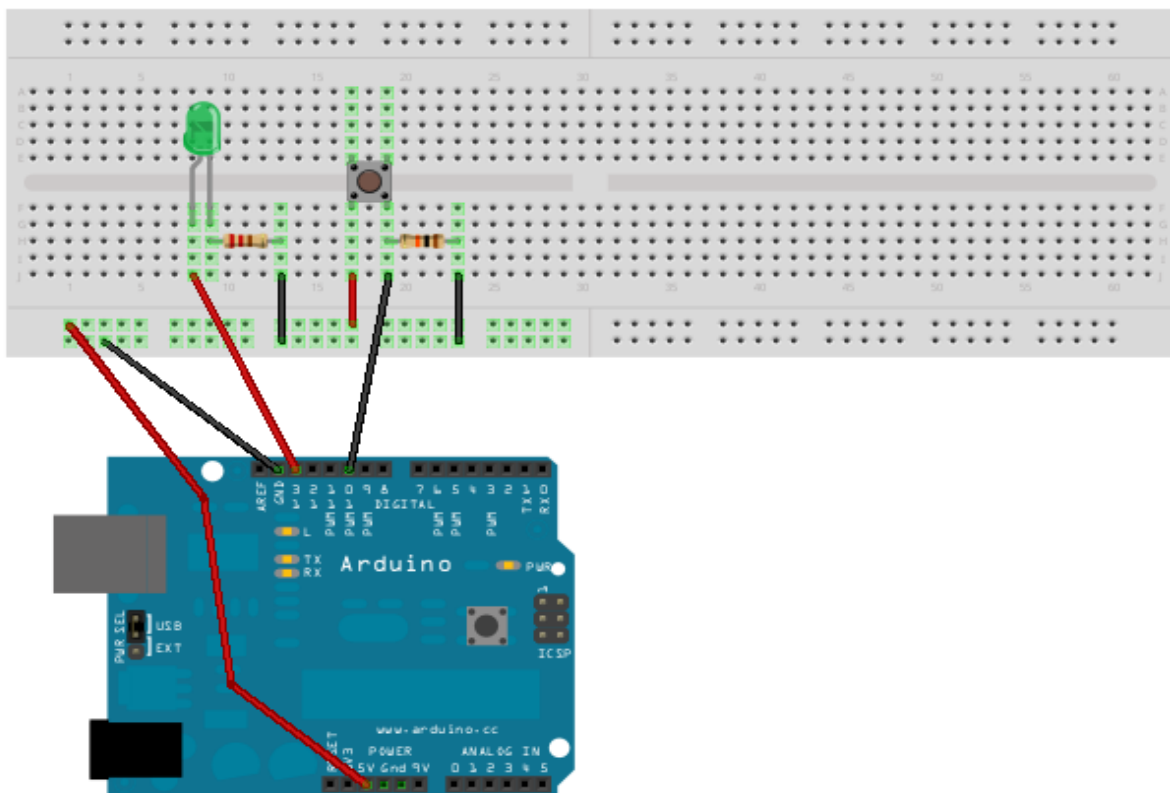
Τώρα θα μελετήσουμε τον τρόπο που μπορούμε να εισάγουμε δεδομένα σε ένα pin. Όπως είδαμε στην έξοδο, έτσι και στην είσοδο η τιμές που μπορεί να δεχτεί ένα pin είναι HIGH ή LOW. Την τιμή HIGH μπορούμε να την πάρουμε από το pin 5V του Arduino ενώ την τιμή LOW μπορούμε να την πάρουμε από τα pin GND (ground, 0V). Για την επιλογή της τιμής 0V ή 5V θα μας βοηθήσει η χρήση ενός διακόπτη.

Παρακάτω θα προσπαθήσουμε με την βοήθεια ενός διακόπτη να ανάβουμε ένα led για ένα δευτερόλεπτο.

Τα υλικά που θα χρειαστούμε σε αυτή την εφαρμογή είναι:

- 1 x breadboard
- 1 x led
- 1 x αντίσταση 220Ω
- 1 x διακόπτη
- 1 x αντίσταση 10kΩ (καφέ – μαύρο – πορτοκαλί)

Η συνδεσμολογία που κάναμε φαίνεται στο παρακάτω σχήμα:



Εικόνα 4: Κύκλωμα τυπικής εισόδου και εξόδου

Το πρόγραμμα που φορτώσαμε στο Arduino είναι το παρακάτω:

```
int ledPin = 13;
int inPin = 10;

void setup(){
  pinMode(ledPin, OUTPUT);
  pinMode(inPin, INPUT);
}

void loop(){
  if (digitalRead(inPin) == HIGH){
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
  }
}
```

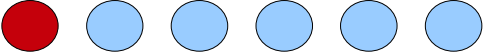





Όπως παρατηρείτε, τις περισσότερες εντολές τις μελετήσαμε στο προηγούμενο παράδειγμα. Αυτό που μένει να μελετήσουμε είναι η συνθήκη *if (digitalRead(inPin) == HIGH){}*.

Η εντολή *digitalRead(inPin)* διαβάζει τη κατάσταση του pin 10 του Arduino. Αν πιάσουμε τον διακόπτη το pin 10 παίρνει τη τιμή HIGH, δηλαδή κλείνει το κύκλωμα με 5V (βλ. Κύκλωμα), αν αφήσουμε τον διακόπτη τότε το pin γειώνετε (GND) άρα έχουμε κατάσταση LOW στο pin. Άρα όπως βλέπουμε πιο πάνω, ο κώδικας που βρίσκετε μέσα στην *if()* θα εκτελεστεί μόλις ανιχνευτούν 5V στο ψηφιακό pin.

Ασκήσεις

Άσκηση 1

Χρησιμοποιώντας 6 leds να υλοποιήσετε μια εφαρμογή η οποία να τα κάνει να αναβοσβήνουν με τη σειρά κάνοντας παύση μισού δευτερολέπτου κάθε φορά. Η σειρά που θα ανάβουν θα είναι η εξής:

Σειρά εκτέλεσης	Κατάσταση led
1	
2	
3	
4	
5	
6	

Στη συνέχεια αν θέλουμε μπορούμε να αλλάξουμε την σειρά εκτέλεσης σε:

1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1

Να χρησιμοποιηθούν τα ψηφιακά pin 2 ως 7 για τα led.

* Σε μελλοντικό στάδιο θα μπορούμε να κάνουμε χρήση βρόγχου ώστε να μειώσουμε το μέγεθος του κώδικα αλλά και να αυτοματοποιούμε την διαδικασία της εκτέλεσης.

Άσκηση 2

Αφού μελετήστε την εντολή `random()` από το reference του Arduino (Help -> Reference) να υλοποιήσετε ένα ηλεκτρονικό ζάρι, χρησιμοποιώντας 6 led και ένα διακόπτη. Κάθε φορά που θα πατάτε τον διακόπτη να παράγεται ένας τυχαίος αριθμός από το 1 ως το 6 και στη συνέχεια να ανάβει το ανάλογο led.

Τα led πρέπει να τοποθετηθούν το ένα δίπλα από το άλλο στη breadboard ώστε το κάθε ένα να αντιστοιχεί σε ένα αριθμό, για το πρώτο led ο αριθμός 1, για το δεύτερο ο αριθμός 2 κ.ο.κ.

Να χρησιμοποιηθούν τα ψηφιακά pin 2 ως 7 για τα led, ενώ ο διακόπτης να τοποθετηθεί στο ψηφιακό pin 10.

Για απορίες μπορείτε να στέλνετε mail στο: ghadjikyriacou@yahoo.com