ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΔΗΜΙΟΥΡΓΊΑ ΠΑΙΧΝΙΔΙΟΥ ΜΕ ARDUINO ΚΑΙ PROCESSING

ΠΤΥΧΙΑΚΉ ΕΡΓΑΣΙΑ

ΤΟΥ

ΚΩΝΣΤΑΝΤΙΝΟΥ ΠΡΑΤΤΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΙΩΑΝΝΗΣ ΚΟΥΡΕΤΑΣ

ΣΠΑΡΤΗ 2016

Ευχαριστίες

Πριν ξεκινήσω θα ήθελα να ευχαριστήσω τους γονείς μου που με στήριξαν όλο αυτό το διάστημα κατά την φοίτηση μου στο ΤΕΙ και όχι μόνο.Επίσεις θα ήθελα να ευχαριστήσω τον καθένα ξεχωριστά αλλά και όλους μαζί τους καθήγητες του τμήματος που αυτά τα χρόνια ήταν δίπλα μου στην προσπάθεια μου για την επίτευξη των στόχων μου.Και φυσικά θα ήθελα να πω ένα μεγάλο ευχαριστώ στον επιβλέπων καθηγητή Ιωάννη Κουρέτα που μου παρείχε όλον τον εξοπλισμό για την υλοποίηση του παιχνιδιού αλλά ήταν και άμεσα διαθέσιμος σε ότι μου προέκυπτε.

Εισαγωγή

Σε αυτήν την πτυχιακή αναλύονται λεπτομερώς η γλώσσα προγραμματισμού processing καθώς και η γλώσσα προγραμμάτισμου wiring που χρησιμοποιείται για τον προγραμματισμό του arduino. Αναλύονται οι βασικές συναρτήσεις όπως και πως μπορεί κάποιος αρχάριος στις συγκεκριμένες γλώσσες να καταφέρει να γράψει το πρώτο του πρόγραμμα.Το κυρίως θέμα και αποκορύφωμα αυτής της πτυχιακής αποτελεί το παιχνίδι που υλοποιήθηκε χρησιμοποιώντας σε συνδυασμό και τις δυο προαναφέρουσες γλώσσες.

Περιεχόμενα

Κεφάλαιο 1

Η ιστορία πίσω από την γλώσσα πραγραμματισμού Processing

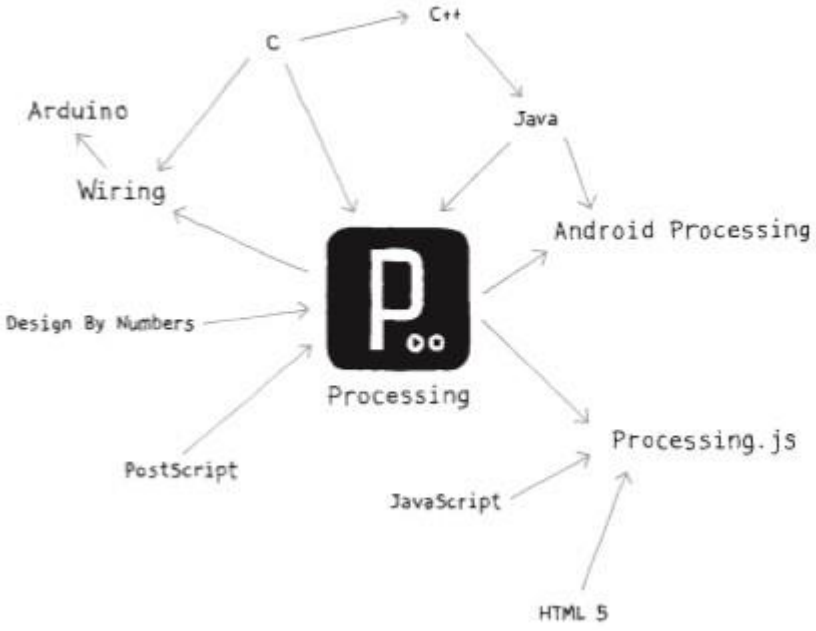
Η γλώσσα προγραμματισμού processing ξεκίνησε,το 2001, απο το πανεπιστήμιο την Μασαχουσέτης MIT όταν 2 σπουδαστές του εμφανίστηκαν στην τάξη της αισθητικής και υπολογισμού,έχοντας συλλάβει μια ιδέα που την ελέγαν processing.Σκοπός τους ήταν να βοηθήσουν άτομα που ασχολούνται με την γραφιστική και δεν είχαν πολλές γνώσεις εως και καθόλου γνώσεις με τον προγραμματισμό.Ηθελαν να δημιουγήσουν μια γλώσσα τόσο απλή στην εκμάθηση σε επίπεδα που πλησίαζε την γλώσσα προγραμματισμού Basic.Οι 2 αυτοί σπουδαστές , **Benjamin Fry** και **Casey Reas**, βλέποντας το μεγάλο ενδιαφέρων που προξένησε η ιδέα του κυκλοφόρησαν την έκδοση alpha το 2002 και στην συνέχεια την έκδοση beta το 2008.Η processing είχε γίνει αναπόσπαστο εργαλείο πολλών προγραμματιστών για την δημιουργία γραφικών περιβάλλοντων ,γραφίστων,αλλά την είχαν υιοθετήσει και πολλά πανεπιστήμια.Έτσι έβγαλαν την πρώτη επίσημη έκδοση 1.0*[β].*

Σήμερα η processing έχει επεκταθεί σε ακόμα μεγαλύτερο κοινό σε όλο τον κόσμο έχοντας δημιουργηθεί για την γλώσσα πάνω απο 100 προσθήκες και βιβλιοθήκες που επεκτείνουν την χρησιμότητα του προγράμματος καθώς και βιβλία για την εκμάθηση και κατανόηση της γλώσσας.Οι δημιουργοί της βλέποντας την επιτυχία της γλώσσας ίδρυσαν το ίδρυμα processing(**processing foundation**) με σκοπό να βοηθήσουν την διάδοση και εκμάθηση της γλώσσας.

Τι είναι Processing

Το processing είναι μια ευέλικτη γλώσσα προγραμματισμού για την δημιουργία εικόνων ,κινούμενων σχεδίων και αλληλεπιδράσεων μεταξύ τους*[ι1]*.Είναι η απλή ιδέα όπου γράφεις μια γραμμή κώδικα και ένα τετράγωνο εμφανίζεται στην οθόνη.Με απλές εντολές μπορείς να του δώσεις χρώμα, κίνηση και αλληλεπίδραση με άλλα αντικείμενα όπως για παράδειγμα την κίνηση ή απλά ένα κλικ του ποντικιού.

Η processing αποτελεί μια πιο απλουστευμένη μορφή της γλώσσας προγραμματισμού java όμως εμπεριέχει διάφορες συναρτήσεις για τα γραφικά και τις αλληλεπιδράσεις τους.Ωστόσο τα γραφικά στοιχεία της γλώσσας είναι συνδεδεμένα με Postscript και OpenGL.Λόγο των ανώτερο αποτελεί ένα πολύ καλό πρώτο βήμα για νέους προγραμματιστές διότι εμπεριέχει στοιχεία από διάφορες γλώσσες.Στην παρακάτω εικόνα φαίνετε το γενεαλογικό δένδρο της γλώσσας*(εικόνα 1.1)*:

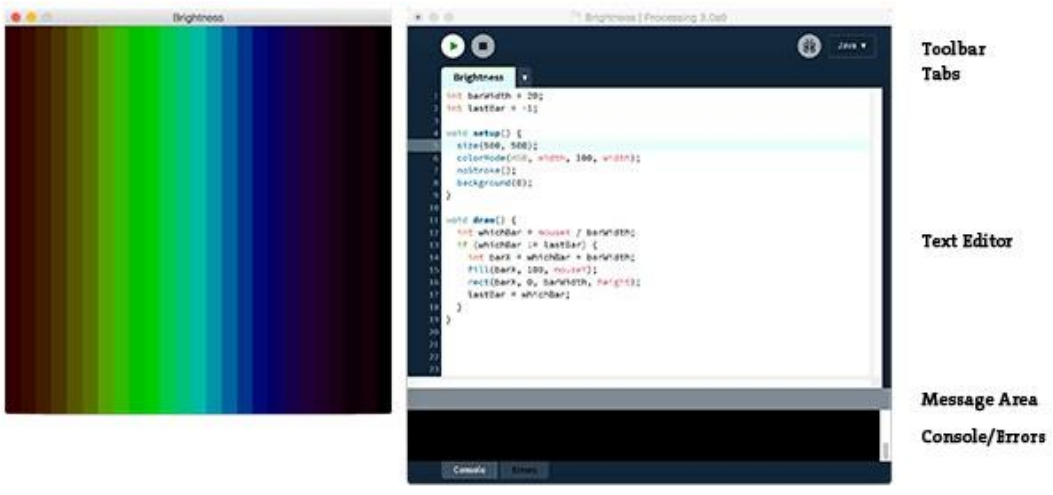


Εικόνα 1.1:Γενεολογικό Δένδρο

Περιβάλλον Δημιουργίας Processing (PDE)

Το **processing development environment (PDE)** αποτελεί το εργαλείο για την δημιουργία προγραμμάτων σε processing.Αποτελείτε από τον text-editor ,το πεδίο μηνυμάτων προς τον προγραμματιστή ,μια μπάρα με επιλογές και το πεδίο εκτέλεσης του προγράμματος*[ι2]*. Τα προγράμματα δημιουργούνται στον text editor που διαθέτει το πρόγραμμα και τρέχουν πατώντας το 'run' στην πάνω αριστερή γωνία τα αποτελέσματα εμφανίζονται στο πεδίο εκτέλεσης*(εικόνα 1.2)*.Τα προγράμματα σε processing ονομάζονται **σκίτσα(sketch)**.Τα σκίτσα μπορούν να έχουν 2 ή 3 διαστάσεις.Το πρόγραμμα από προεπιλογή δημιουργεί 2 διαστάσεις.

Το πεδίο εμφάνισης το καθορίζει ο προγραμματιστής του δίνει τις διαστάσεις κατά την διάρκεια του προγραμματισμού ουσιαστικά αποτελεί τον καμβά όπου πάνω του θα εμφανιστούν τα σκίτσα μας.



Εικόνα 1.2: Processing Development Environment

Πως ξεκινάμε με την processing

Για να ξεκινήσουμε να φτιάχνουμε προγράμματα σε processing δεν αρκεί να ξέρουμε απλά τις εντολές που χρειάζονται για την δημιουργία σχημάτων, χρωμάτων κτλ. Χρειαζόμαστε και ένα πρόγραμμα που μπορεί να καταλάβει τις εντόλες και να μας δώσει το επιθυμητό αποτέλεσμα αυτό το πρόγραμμα μπορούμε να το βρούμε στην επίσημη ιστοσελίδα της processing(www.processing.org). Το πρόγραμμα είναι ανοιχτού κώδικα(open source) δηλαδή παρέχεται εντελώς δωρεάν.Μετά την αντικατάσταση το πρόγραμμα είναι απολύτως έτοιμο για να δημιουργήσουμε τις εφαρμογές μας σε processing.

Δομή του προγράμματος

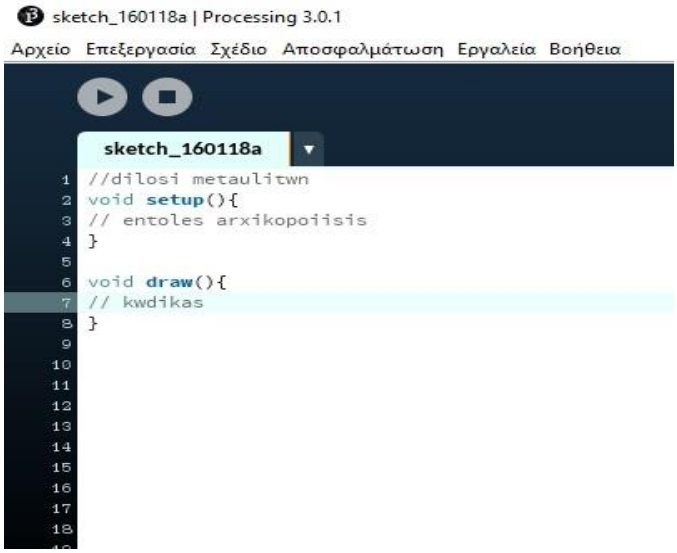
Η processing είναι μια γλώσσα αντικειμενοστραφούς προγραμματισμού δηλαδή οργανώνει τόσο το πρόβλημα όσο και την λύση του σε διακριτά αντικείμενα. Η δομή κάθε προγράμματος στην processing περιέχει δύο στάδια :

1. Δήλωση μεταβλητών
2. Την αρχικοποίηση
3. Το κυρίως πρόγραμμα

Αρχικά δηλώνουμε τις μεταβλητές όπου μπορούμε είτε να τις καταχωρήσουμε τιμές εδώ ή απλά να τις δηλώσουμε και να τις καταχωρήσουμε τιμές στο κυρίως πρόγραμμα.

Στην διαδικασία της αρχικοποίησης ορίζουμε τον καμβά που θα εμφανίζεται το πρόγραμμά μας καθώς και τιμές οι οποίες δεν θα επηρεάζονται κατά την εκτέλεση του προγράμματος.Η αρχικοποίηση ορίζεται με το **void setup() {..}** όπου μέσα στις αγκύλες μπαίνουν οι εντολές αρχικοποίησης.

Στο κυρίως πρόγραμμα περιέχονται εντολές δημιουργίας σχημάτων,χρωματων,δομών,διαδικασιών κ.α. Το ορίζουμε με **void draw(){…}**.Έτσι ένα πρόγραμμα έχει την δομή που φαίνεται παρακάτω(εικόνα 1.3):



Εικόνα 1.3:Δομή Προγράμματος

Δημιουργία καμβά

Για να φτίαχνουμε τον χώρο όπου θα εμφανίζεται το πρόγραμμα μας θα πρέπει να το ορίσουμε στην αρχικοποίηση των τιμών με την εντολή size(_,_).Έχει δύο παραμέτρους την παράμετρο χ και την παράμετρο ψ.Οι δύο αριθμοί που θα δώσουμε θα καθορίσουν το μέγεθος του καμβά μας.Η παράμετρος χ καθορίζει το πλάτος του καμβά ενώ το ψ καθορίζει το ύψος του καμβά.Για παράδειγμα όταν έχουμε χ και ψ 100 τότε θα έχουμε έναν τετράγωνο καμβά ενώ χ ίσο με 500 και ψ ίσο με 200 θα έχουμε ορθογώνιο καμβά(εικόνα 1.4 και 1.5).

Εικόνα 1.4.: Τετράγωνος Καμβάς

Εικόνα 1.5:Ορθογώνιος Καμβάς

Αντίστοιχα για να σχεδιάσουμε γραφικά τριών διαστάσεων(3D) θα πρέπει να αρχικοποιήσουμε και τον κατάλληλο τριών διαστάσεων καμβά και αυτό γίνεται με την εντολή **size(_,_,3D)**.Άμα προβάλουμε ένα καμβά 2 διαστάσεων και έναν 3 διαστάσεων δεν έχουν κάποια εμφανή διαφορά απλά διαφέρουν στα γραφικά που θα σχεδιαστούν πάνω τους.

Βασικά σχέδια στην processing

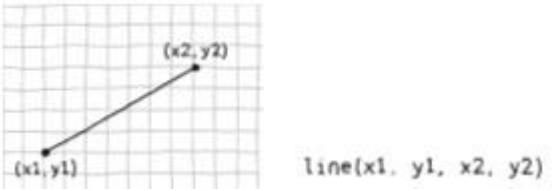
Τα βασικά σχέδια σε processing είναι αυτά όπου έχουν την δικιά τους συνάρτηση καταχωρημένη στην γλώσσα processing για τις 2 διαστάσεις είναι:

- Ένα σημείο
- Η ευθεία
- Το τετράπλευρο

- Το ορθογώνιο παραλληλόγραμμο
- Η έλλειψη
- Μέρη από έλλειψη

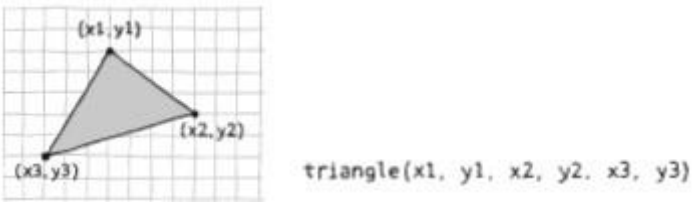
Το σημείο ορίζεται με την συνάρτηση **point(x,y)** και έχει 2 παραμέτρους την παράμετρο χ και την παράμετρο ψ όπου καθορίζουν το πού θα εμφανιστεί το σημείο.

Η ευθεία ορίζεται από την συνάρτηση **line(x1,y1,x2,y2)**.Για το σχεδιασμό της ευθείας χρειαζόμαστε δύο σημεία της, την αρχή και το τέλος της.Έτσι σε αυτήν την συνάρτηση το χ1 και το ψ1 αντιπροσωπεύουν τις συντεταγμένες του σημείου αρχής και το χ2-ψ2 τις συντεταγμένες του σημείου του τέλους(εικόνα 1.6).



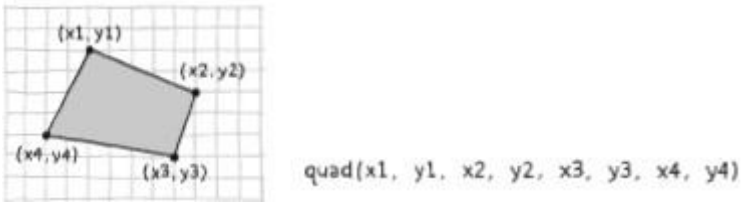
Εικόνα 1.6:Συνάρτηση Γραμμή

Το τρίγωνο ορίζεται με την συνάρτηση **triangle(x1,y1,x2,y2,x3,y3)**. Για να σχεδιάσουμε ένα τρίγωνο χρειαζόμαστε τρία σημεία όσες και οι κορυφές του τριγώνου.Το χ1-ψ1,χ2-ψ2 και χ3-ψ3 αντιπροσωπεύουν τις συντεταγμένες για κάθε μια από τις κορυφές του τριγώνου(εικόνα 1.7).



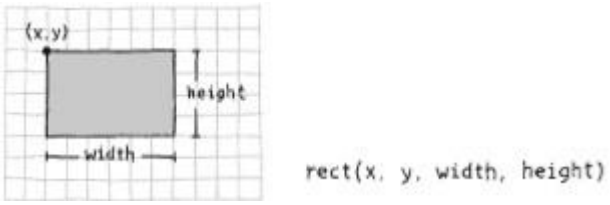
Εικόνα 1.7:Συναρτήση Τριγώνου

Το τετράπλευρο δημιουργείται από την συνάρτηση **quad(x1,y1,x2,y2,x3,y3,x4,y4)** .Όπως και στο τρίγωνο στο τετράπλευρο χρειαζόμαστε 4 κορυφές.Τα χ και τα ψ αντιπροσωπεύουν τις συντεταγμένες για κάθε μια από τις κορυφές του(εικόνα 1.8).



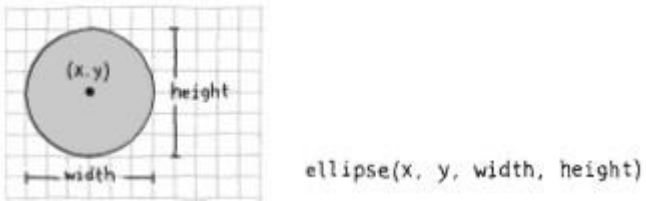
Εικόνα 1.8:Συνάρτηση Τετράπλευρου

Το ορθογώνιο παραλληλόγραμμο ορίζεται από την συνάρτηση **rekt(x,y,width,height)**.Για να φτιάξουμε ένα παραλληλόγραμμο χρειαζόμαστε μια κορυφή καθώς το ύψος και το πλάτος του.Το χ και το ψ ορίζει την πάνω αριστερά κορυφή το πλάτος και το ύψος του(εικόνα 1.9).



Εικόνα 1.9:Συνάρτηση Ορθογωνίου

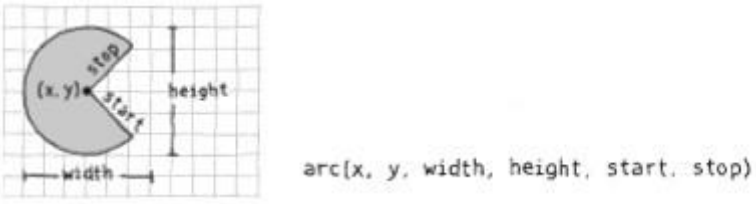
Για να σχηματίσουμε την έλλειψη η συνάρτηση είναι **ellipse(x,y,width,height)**. Για την δημιουργία της έλλειψης χρειαζόμαστε το κέντρο της έλλειψης το πλάτος και το ύψος της(εικόνα 1.10).Δεν υπάρχει συνάρτηση που να ορίζει κύκλο μπορούμε όμως να τον ορίσουμε δίνοντας ίσο πλάτος με το ύψος.



Εικόνα 1.10:Συνάρτηση Έλλειψης

Η processing μας δίνει την δυνατότητα να δημιουργήσουμε ακόμη και ένα μέρος της έλλειψης με την συνάρτηση **arc(x,y,width,height,start,stop)**. Ισχύουν τα ίδια με την έλλειψη το μόνο

που αλλάζει είναι ότι πρέπει να δώσουμε την αρχή και το τέλος του ημικύκλιου σε μοίρες(εικόνα 1.11).



Εικόνα 1.11:Συνάρτηση Ημικύκλιου

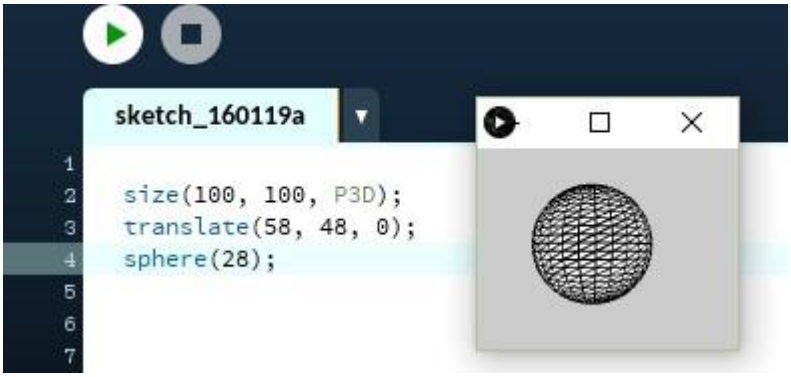
Υπάρχουν και σχήματα και για τις 3 διαστάσεις αυτά είναι:

- Ο Κύβος
- Η σφαίρα

Ο κύβος σχηματίζεται από την συνάρτηση **box(x,y,z)**.Το χ συμβολίζει το ύψος,το ψ το πλάτος και το z το μήκος.Στα σχήματα τριών διαστάσεων ο σχηματισμός του σχήματος ξεκινάει από το σημείο ένωσης των αξόνων το σημείο 0.Γι'αυτόν τον λόγο μαζί με τα σχήματα με 3 διαστάσεις ορίζουμε άλλη μια συνάρτηση η οποία ορίζει το πού θα σχηματιστεί το τριών διαστάσεων σχήμα μας.Αυτή η συνάρτηση είναι η **translate(x,y,z)**.Επίσης την συνάρτηση της περιστροφής που είναι rotateX(x) ή rotateY(y) ή rotate(z) αναλόγως με το που θέλουμε να την περιστρέψουμε(εικόνα 1.12).

Εικόνα 1.12:Παράδειγμα Συνάρτησης Κουτιού

Η σφαίρα ορίζεται από την συνάρτηση **sphere(r)**.Το μόνο που χρειαζόμαστε για να ορίσουμε την σφαίρα είναι η ακτίνα της(εικόνα 1.13).



Εικόνα 1.13:Παράδειγμα Συνάρτησης Σφαίρας

Πως γεμίζουμε με χρώμα την processing

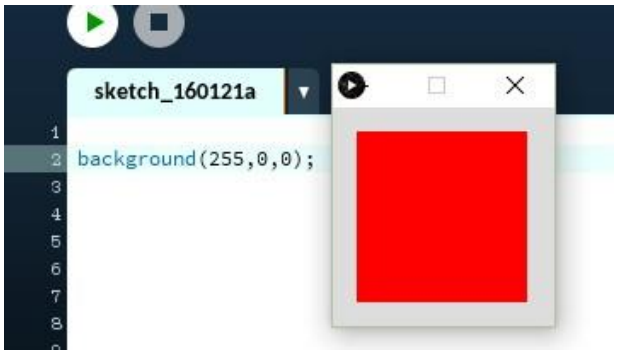
Η processing μας δίνει την δυνατότητα με δύο βασικές συναρτήσεις να προσθέσουμε χρώμα :

- Στο background
- Στα σχήματά μας

Για να δώσουμε χρώμα στο background θα πρέπει να χρησιμοποιήσουμε την συνάρτηση **background()**.Η συγκεκριμένη συνάρτηση μπορεί να χρησιμοποιηθεί με 2 τρόπους αναλόγως με το χρώμα που θέλουμε να δώσουμε.Για λευκό ή μαύρο χρώμα αλλά και για τις μεταξύ τους αποχρώσεις χρησιμοποιούμε την απλή μορφή της συνάρτησης που έχει ένα όρισμα το οποίο μπορεί να έχει τιμές από 0 έως 255.Με το 0 ορίζεται το μαύρο και με το 255 ορίζεται το άσπρο,εάν δώσουμε έναν άλλο αριθμό θα αποτελεί μια μεταξύ τους απόχρωση(εικόνα 1.14).

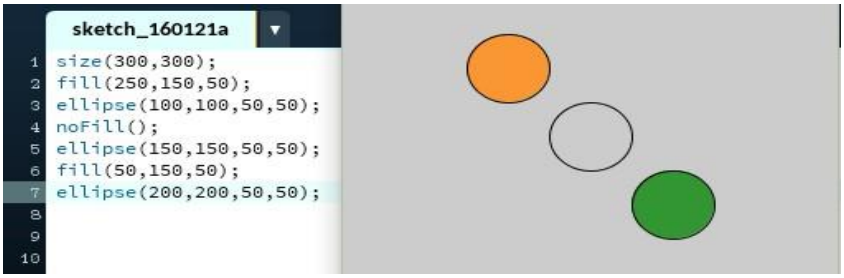
Εικόνα 1.14:Παράδειγμα Συνάρτηση Φόντου

Για να δώσουμε στην συνάρτηση κάποιο άλλο χρώμα χρησιμοποιούμε την συνάρτηση με 3 ορίσματα που το κάθε όρισμα παίρνει τιμές από 0 έως 255.Η πιο σύνθετη μορφή της συνάρτησης χρησιμοποιεί το μοντέλο χρωμάτων RGB(red-green-blue) δηλαδή η τελική απόχρωση του background καθορίζεται από το πόσο έντονους τόνους έχουμε δώσει στα 3 ορίσματα.Άμα για παράδειγμα θέλουμε να φτιάξουμε ένα background κόκκινο θα ορίσουμε το την συνάρτηση με την εξής μορφή background(255,0,0) όπως φαίνεται στην εικόνα(εικόνα 1.15).



Εικόνα 1.15: Παράδειγμα Συνάρτηση Φόντου

Μόλις θέλουμε να δώσουμε χρώμα σε αντικείμενα χρησιμοποιούμε την συνάρτηση **fill()**. Και αυτή η συνάρτηση έχει 2 μορφές με 1 όρισμα ή με 3 ορίσματα και ισχύει ότι ακριβώς και στην συνάρτηση background().Για να ορίσουμε ένα σχήμα με χρώμα θα πρέπει να βάλουμε την συνάρτηση πάνω από σχήμα.Αυτή η συνάρτηση ορίζει το χρώμα όλων των σχημάτων που υπάρχουν στις επόμενες γραμμές του κώδικα.Σε περίπτωση που θέλουμε να χρωματίσουμε ένα άλλο σχήμα διαφορετικό χρώμα θα πρέπει να βάλουμε μια καινούργια συνάρτηση fill() από πάνω του.Επίσεις άμα δεν θέλουμε να χρωματίσουμε κάποιο σχήμα θα πρέπει να βάλουμε πάνω από το σχήμα την συνάρτηση noFill() η οποία δεν παίρνει ορίσματα(εικόνα 1.16).



Εικόνα 1.16:Παράδειγμα Συνάρτησης Γεμίσματος

Μεταβλητές στην processing

Όπως σε όλες τις γλώσσες προγραμματισμού έτσι και στην processing μπορούμε να έχουμε μεταβλητές.Οι μεταβλητές αναλόγως με τον τύπο τους δηλώνονται και διαφορετικά μέσα στο πρόγραμμα.Μπορούμε να τις δηλώσουμε είτε πριν την συνάρτηση void setup() είτε κατά την εκτέλεση του προγράμματος.Οι μεταβλητές μπορεί να είναι ακαίρεοι,δεκαδικοί,χαρακτήρες και πολλές άλλες μορφές.Στις μεταβλητές μπορούμε να δώσουμε απευθείας τιμές ή να της χρησιμοποιήσουμε κατά την διάρκεια του προγράμματος και να τους εκχωρήσουμε τιμές.Οι ακέραιες μεταβλητές δηλώνονται με το **int** οι δεκαδικές μεταβλητές με **float** και οι χαρακτήρες με **char**. Για παράδειγμα εάν θέλουμε να φτιάξουμε μια μεταβλητή ακέραια με όνομα α και τιμή 2 θα πρέπει να γράψουμε το εξής int a=2; .Το ελληνικό ερωτηματικό συμβολίζει το τέλος της γραμμής και θα πρέπει να μπαίνει σχεδόν σε όλες τις γραμμές.Μολις εκχωρούμε τιμή σε μια μεταβλητή η τιμή αυτή δεν είναι μόνιμη μπορεί να αλλάξει κατά την εκτέλεση του προγράμματος.Στην παρακάτω εικόνα το τελικό αποτέλεσμα της μεταβλητής α είναι 8 (εικόνα 1.17).

Εικόνα 1.17:Παράδειγμα ανάθεσης τιμής σε μεταβλητή

Επίσης με τις μεταβλητές μπορούμε να κάνουμε πράξεις για τον υπολογισμό του τελικού αποτελέσματος της.Η processing δέχεται όλα τα αριθμητικά σύμβολα πράξεων.Για παράδειγμα η μεταβλητή α θα μπορούσε να είναι το αποτέλεσμα κάποιον πολλαπλασιασμών,διεράσεων και προσθέσεων (εικόνα 1.18).

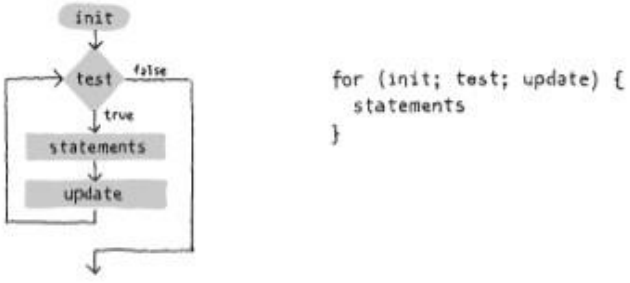
Εικόνα 1.18:Παράδειγμα Ανάθεσης Μεταβλητών

Δομές επανάληψης

Μόλις θέλουμε να αναπαράγουμε ένα κομμάτι κώδικα 2 ή περισσότερες φορές χρησιμοποιούμε τις δομές επανάληψης.Οι δύο δομές επανάληψης που υποστηρίζονται στην processing είναι:

- for()
- while()

Η δομή επανάληψης for παίρνει 3 ορίσματα το πρώτο όρισμα είναι εκεί που ξεκινάει η συνάρτηση για παράδειγμα μια μεταβλητή χ που είναι ίση με το 0.Το δεύτερο όρισμα είναι εκεί όπου θα τελειώνει η δομή επανάληψης για παράδειγμα η μεταβλητή χ <10.Έτσι μέχρι αυτή η συνθήκη να μην ισχύει θα τρέχει η δομή.Και στο τελευταίο όρισμα κατά πόσο θα αυξάνει η μεταβλητή για παράδειγμα χ=χ+1(εικόνα 1.19).



Εικόνα 1.19:Παράδειγμα Δομής Επανάληψης FOR

Η δομή επανάληψης while αποτελείτε από μόνο ένα όρισμα που είναι μια συνθήκη που για όσο είναι αληθείς θα τρέχει ο κώδικας. Η διαφορά με το for() είναι ότι πρέπει να ορίσουμε μια μεταβλητή έξω από την δομή επανάληψης και να την αυξάνουμε μέσα στην δομή επανάληψης (εικόνα 1.20).

```
int i = 0;
while (i < 80) {
  line(30, i, 80, i);
  i = i + 5;
}
```

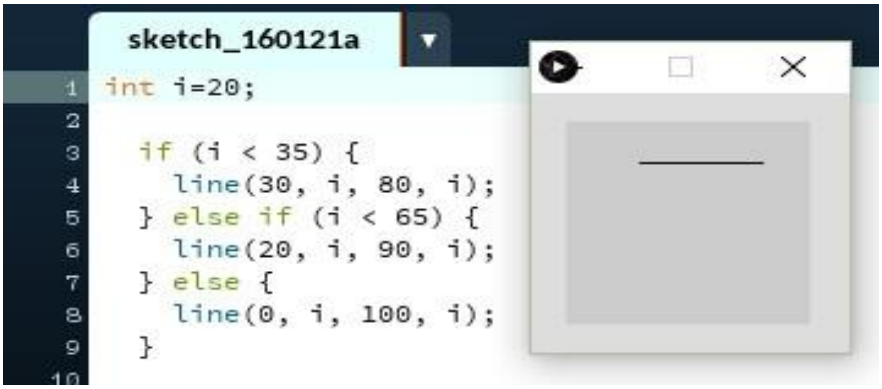
Εικόνα 1.20:Παράδειγμα Δομής Επανάληψης while

Δομές Επιλογής

Όταν θέλουμε να εκτελέσουμε κώδικα που εξαρτάται από διαφορές μεταβλητές ή αποτελέσματα χρησιμοποιούμε τις δομές επιλογής.Στην processing έχουμε δύο δομές επιλογής αυτές είναι:

- **If…else**
- **Switch**

Στην δομή επιλογής **if** μπορούμε να έχουμε όσα ορίσματα θέλουμε και αυτό εξαρτάται από την συνθήκη που θέτουμε. Η συνθήκη μπαίνει μέσα σε παρενθέσεις μετά το if και αποτελείτε από συγκρίσεις μεταξύ μεταβλητών δηλαδή μέγαλύτερης,μικρότερης,ίσης αξίας ή διαφορετικά μεταξύ τους.Μεταξύ τους οι συγκρίσεις ενώνονται με το **and** ή το **or** που συμβολίζονται με **&&** και **||** αντίστοιχα.Μόλις βάζουμε το and οι συνθήκες που ενώνονται πρέπει να ισχύουν και οι δύο αλλιώς δεν εκτελείται η συνθήκη επιλογής.Το or χρησιμοποιούμε όταν θέλουμε να ισχύει τουλάχιστον μια από τις δύο συνθήκη. Όταν θέλουμε να κάνουμε επιλογή κώδικα που θα εκτελείτε στην περίπτωση που δεν εκτελείτε η προηγούμενη συνθήκη τότε χρησιμοποιούμε το **else**.Στην περίπτωση που έχουμε παραπάνω από δύο τέτοιες περιπτώσεις τότε χρησιμοποιούμε το else if το οποίο ακολουθείται και πάλι από μια συνθήκη με συγκρίσεις (εικόνα 1.21).



Εικόνα 1.21:Παράδειγμα Δομής Επιλογής if

Η δομή επιλογής **switch** συντάσσεται με μόνο ένα όρισμα το οποίο εκτελεί ανάλογα με το αποτέλεσμα και έναν διαφορετικό κώδικα.Η κάθε πιθανότητα κώδικα μπαίνει μετά το **case** και ακολουθεί το αποτέλεσμα της μεταβλητής που εξετάζουμε έπειτα άνω και κάτω τελεία και στην επόμενη γραμμή ο κωδικας.Μετά το τέλος του κώδικα πρέπει να βάλουμε την λέξη **break** που του ορίζει το τέλος της δομής επιλογής.Για παράδειγμα άμα είχαμε την επιλογή να ζωγραφίσουμε έναν κύκλο και είχαμε 3 επιλογές πράσινο,κόκκινο και μπλε ο παρακάτω κώδικας θα δημιουργούσε τον κύκλο ανάλογα με την επιλογή μας (εικόνα 1.22).

```
switch(e)
{
 case k:
      fill(255,0,0);
      ellipse(50,50,50,50);
    break;
 case g:
      fill(0,255,0);
      ellipse(50,50,50,50);
    break;
 case b:
     fill(0,0,255);
     ellipse(50,50,50,50);
    break;
}
```

Εικόνα 1.22: Παράδειγμα Δομής Επιλογής switch

Δημιουργία πίνακα

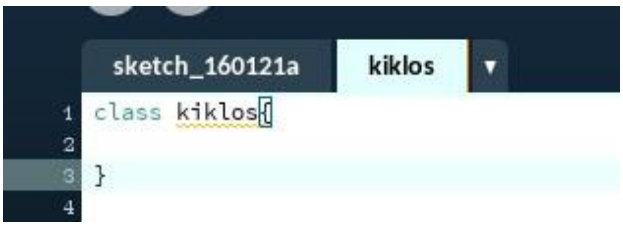
Ένας πίνακας μπορεί να αποτελείτε από οποιοδήποτε τύπο μεταβλητής ή αντικειμένου.Όπως και στις μεταβλητές μπορούμε να καταχωρήσουμε τις τιμές του πίνακα στην δημιουργεία του ή να τις καταχωρήσουμε κατά την εκτέλεση του προγράμματος.Ο πίνακας ορίζεται από τον τύπο της μεταβλητής ή του αντικειμένου έπειτα τις 2 τετράγωνες αγκύλες **[]** και το όνομα του πίνακα μετά ακολουθεί το ίσον (=) new τον τύπο της μεταβλητής και μέσα σε τετράγωνες αγκύλες τον αριθμό των κελιών που θα αποτελείτε ο πίνακας .Για παράδειγμα άμα θέλουμε να δημιουργήσουμε έναν πίνακα ακέραιων αριθμών με δύο κελία θα γράψουμε το εξής: **int[] x = new int[2];**. Για να καταχωρίσουμε περιεχόμενο στα κελιά θα πρέπει να γράψουμε το όνομα του πίνακα μέσα σε τετράγωνες τον αριθμό του κελιού το ίσον και την τιμή που θέλουμε να καταχωρήσουμε.Αν για παράδειγμα θέλουμε να καταχωρήσουμε στο πρώτο κελί του πίνακα α την τιμή 9 θα γράφαμε: **α[0]=9;**.Θα πρέπει να σημειώσουμε ότι η αρίθμηση του πίνακα ξεκινά από το 0 και όχι από το 1.

Κλάσεις και δημιουργία αντικειμένων

Οι κλάσεις όπως και σε κάθε άλλη γλώσσα αντικειμενοστραφούς προγραμματισμού είναι ο τρόπος για να ορίσουμε τα αντικείμενα.Κλάση είναι ένας ορισμός που ορίζει τις ιδιότητες και τις μεθόδους του αντικειμένου.Ουσιαστικά μια κλάση αντιπροσωπεύει ένα αντικείμενο στον φυσικό κόσμο.Για παράδειγμα μια κλάση θα μπορούσε να είναι η γάτα.Για να φτιάξουμε και να ολοκληρώσουμε μια κλάση θα πρέπει να ακολουθήσουμε τα εξής βήματα:

1. Δημιουργία της κλάσης
2. Εισαγωγή χαρακτηριστικών
3. Δημιουργία ενός **constructor** για την καταχώριση τιμών στις μεταβλητές
4. Να προσθέσουμε τις μεθόδους

Για να δημιουργήσουμε την κλάση πρέπει να δημιουργήσουμε μια νέα καρτέλα στο περιβάλλον της processing που θα έχει το όνομα της κλάσης για παράδειγμα αν φτιάχναμε μια κλάση για τον κύκλο θα ονομάζαμε την κλάση kiklos.Στην συνέχεια θα πρέπει να φτιάχνουμε την κλάση μέσα στην νέα καρτέλα όπως φαίνεται στην παρακάτω εικόνα (εικόνα 1.23).



Εικόνα 1.23:Παράδειγμα Κλάσης

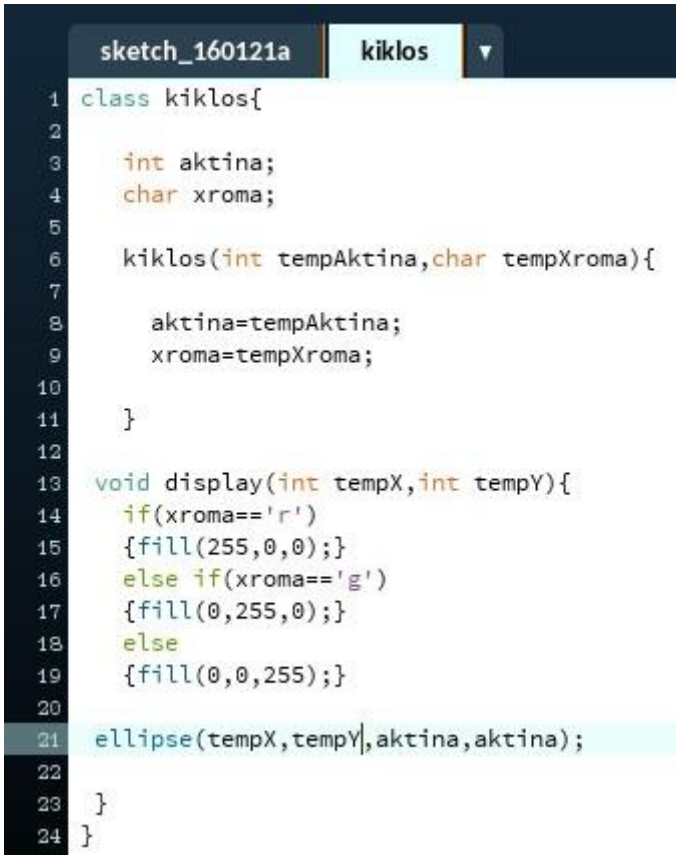
Τα χαρακτηριστικά (attributes) δεν είναι άλλο από τις μεταβλητές που θα ορίζουν το αντικείμενο.Για παράδειγμα ο κύκλος μπορεί να έχει ως χαρακτηριστικά την ακτίνα του και το χρώμα του (εικόνα 1.24).

Εικόνα 1.24:Μεταβλητές Σε Κλάση

Ο constructor είναι μια συνάρτηση που καταχωρεί τις τιμές στα χαρακτηριστικά του αντικειμένου.Αυτή η συνάρτηση θα έχει το ίδιο όνομα με την κλάση, θα έχει προσωρινές μεταβλητές που θα δίνει ο χρήστης και θα καταχωρούνται στα χαρακτηριστικά του αντικειμένου (εικόνα 1.25).

Εικόνα 1.25:Παράδειγμα Constructor

Οι μέθοδοι είναι οι πράξεις των αντικειμένων. Στο αυτοκίνητο μια μέθοδος του είναι η κίνηση του. Η πιο κοινή μέθοδος είναι η μέθοδος της εμφάνισης του αντικειμένου έτσι στο παράδειγμα με τον κύκλο θα ήταν ως εξής(εικόνα 1.26):

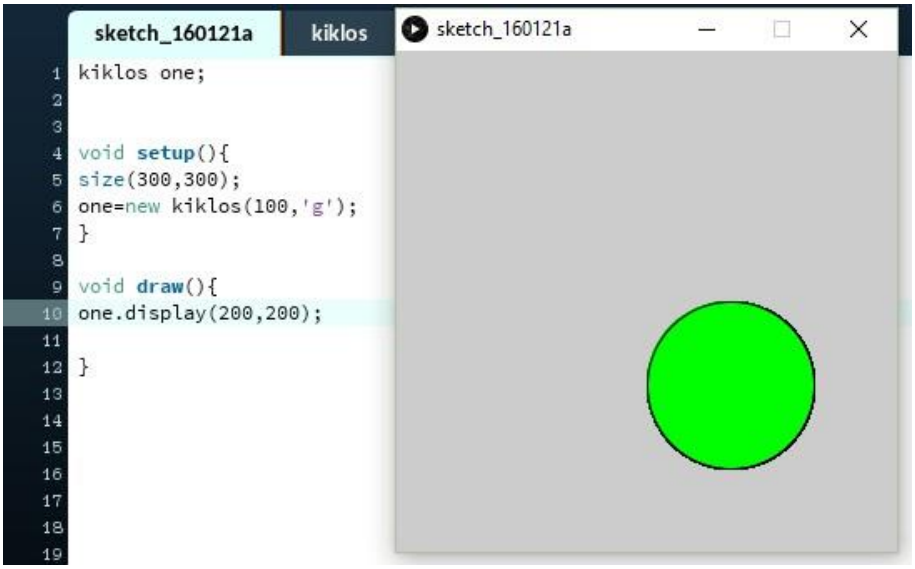


Εικόνα 1.26:Παράδειγμα Μεθόδου

Η παραπάνω μέθοδος ζητάει από τον χρήστη τις συντεταγμένες που θέλει να δημιουργήσει τον κύκλο και περιέχει και μια δομή επιλογής

που καθορίζει το χρώμα του κύκλου (κόκκινο,πράσινο,μπλέ).Στην συνέχεια δημιουργεί τον κύκλο στις συντεταγμένες που έδωσε ο χρήστης.

Για να δημιουργήσουμε αντικείμενα θα πρέπει στο κύριο πρόγραμμα να το δηλώσουμε σαν να ήταν μεταβλητή.Έστω ότι θέλουμε να δημιουργήσουμε κύκλο θα δηλώναμε το όνομα της κλάσης και το όνομα που θα δίναμε στο αντικείμενο για παράδειγμα one έτσι θα έπρεπε να γράψουμε **kiklos one;**. Θα το αρχικοποιήσουμε στην συνάρτηση setup ως εξής: **one=new kiklos(100,'g');** .Για να καλέσουμε τις μεθόδους του αντικειμένου θα πρέπει να δώσουμε το όνομα του αντικειμένου που δημιουργήσαμε θα ακολουθείτε από τελεία και το όνομα της μεθόδου (εικόνα 1.27).



Εικόνα 1.27:Παράδειγμα Δημιουργίας Αντικειμένου

Δημιουργήθηκε ένας πράσινος κύκλος ,γιατί δώσαμε το γράμμα g στην αρχικοποίηση,στις συντεταγμένες 200,200 του καμβά μας όπου δώσαμε στην συνάρτηση display.

Αλληλεπίδραση μπάλας και χρήστη

Σε αυτό το παράδειγμα θα δημιουργήσουμε μια μπάλα που θα κινείτε στην οθόνη.Η μπάλα μόλις θα ακουμπάει στα τοιχώματα του καμβά θα αναπηδά και δεν θα βγαίνει από την οθόνη.Ο χρήστης θα έχει την δυνατότητα να πατήσει το κλικ του ποντικιού δίνοντας μια άλλη τυχαία θέση στην μπάλα.Επίσεις μόλις ο κέρσορας του ποντικιού ακουμπάει την μπάλα αυτή θα αλλάζει χρώματα.Για την υλοποίηση αυτού του παραδείγματος θα πρέπει:

1. Να δημιουργήσουμε τις μεταβλητές .
2. Να δημιουργήσουμε την συνάρτηση setup() και να αρχικοποιήσουμε τις τιμές του καμβά και της μπάλας.
3. Να υλοποιήσουμε την συνάρτηση draw() όπου έχει θα υπάρχει ο κώδικας για το χρώμα του background η αναπήδηση της μπάλας και η αλλαγή του χρώματος της μπάλας.
4. Τέλος θα πρέπει να δημιουργήσουμε μια συνάρτηση για την τυχαία εμφάνιση της μπάλας.

Για την τυχαία εμφάνιση της μπάλας θα χρησιμοποιήσουμε μια συνάρτηση την **random**() όπου παίρνει δύο ορίσματα που ορίζουν την κατώτατη και την ανώτατη τιμή που μπορεί να πάρει.Παρακάτω φαίνεται ο κώδικας του παραδείγματος (εικόνα 1.28 και 1.29).

sketch_160202a

```
float x;
float y;
float xspeed = 1;
float yspeed = 2;

void setup()
  {
  size (500,500);
  noStroke();
  x = width / 2;
  y = height / 2;
  }

void draw()
  {
  background (150);
  if (x >= 490)
    {
    xspeed = -1;
    }
  if (x < 20)
    {
    xspeed = 1;
    }
  if (y >= 490)
    {
    yspeed = -1;
    }
```

Εικόνα 1.28:Υλοποίηση προγράμματος (α μέρος)

```
if (y < 20)
  {
  yspeed = 1;
  }
if (mouseX > x && mouseX < x+30 && mouseY > y && mouseY < y+30)
  {
  fill(random(255), random(255), random(255));
  }
x = x + xspeed;
y = y + yspeed;
ellipse(x,y,30,30);
}

void mousePressed()
{
x = random(10,490);
y = random(10,490);
}
```

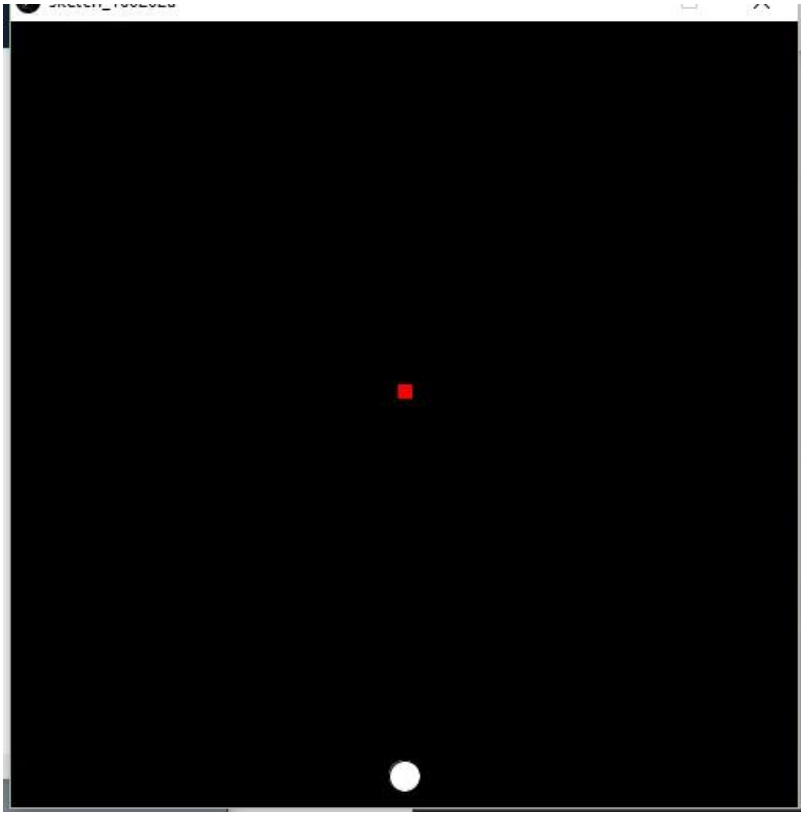
Εικόνα 1.29: Υλοποίηση προγράμματος (β μέρος)

Δημιουργία γραφικού που πυροβολεί

Στο παράδειγμα αυτό θα δημιουργήσουμε έναν κύκλο όπου ο χρήστης θα μπορεί να τον μετακινεί στην οθόνη αριστερά δεξιά και θα έχει την δυνατότητα να πυροβολει.Τα σχήματα που θα χρειαστούμε είναι ένας κύκλος που θα παίζει τον ρόλο του χαρακτήρα μας και ένα ορθογώνιο όπου θα παίζει τον ρόλο της σφαίρας.Αρχικά ορίζουμε το χ και το ψ του κύκλου και του ορθογωνίου.Μετά υλοποιούμε την συνάρτηση setup() όπου θα ορίσουμε τον καμβά μας καθώς επίσης και τις συναρτήσεις ellipsemode () και rektmode() για να κεντράρουμε τα 2 σχήματα μας μαζί.Έπειτα θα ορίσουμε την συνάρτηση draw() όπου θα περιέχει την δημιουργία των δύο σχημάτων ,την κίνηση της σφαίρας και το χρώμα του background.Στο τέλος θα ορίσουμε άλλη μια συνάρτηση που θα ορίζει τα πλήκτρα όπου θα μετακινείται αριστερά και δεξιά ο κύκλος καθώς και το πλήκτρο που θα πυροδοτεί την σφαίρα.Παρακάτω βλέπουμε τον κώδικα δημιουργίας του παραδείγματος και ένα στιγμιότυπο από το αποτέλεσμα (εικόνα 1.30 και 1.31).

```
float x = 250;
float y = 480;
float bullet_x = 250;
float bullet_y = 480;
void setup()
  {
  size (500,500);
  ellipseMode(CENTER);
  rectMode(CENTER);
  smooth();
  }
void draw()
  {
  background (0);
  bullet_y = bullet_y - 5;
  fill(255,0,0);
  rect(bullet_x, bullet_y, 10, 10);
  fill(255);
  ellipse (x,y, 20, 20);
  }
void keyPressed()
  {
  if (key == 'a'){x = x - 5;}
  if (key == 'd'){x = x + 5;}
  if (key == 'p')
  {bullet_y = y;
   bullet_x = x;
    }
```

Εικόνα 1.30:Υλοποίηση Προγράμματος



Εικόνα 1.31:Στοιχμιότυπο Προγράμματος

Δημιουργία ορθογωνίου με κλικ του ποντικιού

Στο παράδειγμα αυτό θα φτιάξουμε μια εφαρμογή όπου θα δημιουργεί ορθογώνια στον καμβά μας με το κλικ του ποντικιού.Για την δημιουργία του θα χρειαστούμε την υλοποίηση της συνάρτησης setup() με την αρχικοποίηση του καμβά και τον καθορισμό των ενεργειών που θα γίνονται στο κέντρο.Θα φτιάξουμε την συνάρτηση draw() όπου δεν θα περιέχει κάτι θα είναι κενή.Και άλλη μια συνάρτηση την mousepress() όπου θα καθορίζει την δημιουργία ενός ορθογωνίου με τυχαίο χρώμα.Τα αποτελέσματα και ο κώδικας φαίνονται στις παρακάτω εικόνες (εικόνα 1.32 και 1.33):

```
1  void setup()
2    {
3    size (500,500);
4    rectMode(CENTER);
5    }
6
7  void draw(){}
8
9  void mousePressed()
10   {
11   fill (random(0,255), random(0,255), random(0,255), 50);
12   rect(mouseX, mouseY, 100,100);
13   }
14
15
```

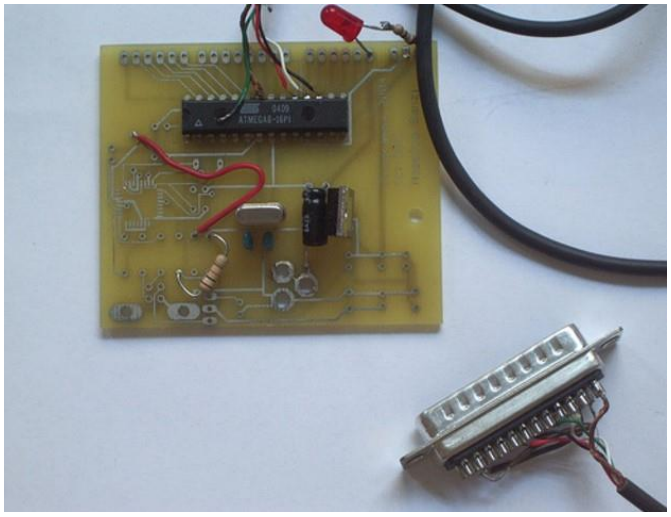
Εικόνα 1.32:Υλοποίηση Προγράμματος

Εικόνα 1.33:Στιγμιότυπο Από Την Υλοποίηση

Κεφάλαιο 2

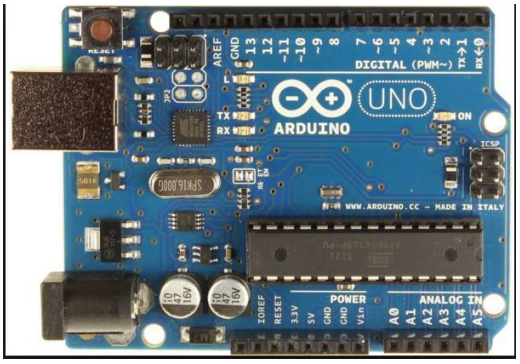
Η ιστορία του arduino

Η ιδέα ξεκίνησε το 2005 στην πόλη Λιβρέα της Ιταλίας,που βρίσκεται κοντά στο Τορίνο, από τους Massimo Banzi και David Cueartielles ως την δημιουργία μιας διαδραστικής πλακέτας όπου θα βοηθούσε τα έργα των μαθητών και δεν θα ήταν τόσο ακριβή όσο άλλα πρωτότυπα κυκλώματα.Η πρωτότυπη πλακέτα που δημιουργήθηκε δεν έφερε το όνομα arduino και είχε την παρακάτω μορφή (εικόνα 2.1):



Εικόνα 2.1:Αρχικό Κύκλωμα Arduino

Το όνομα arduino το πήρε αργότερα το 2005 από τον έναν συνιδρυτή της ιδέας τον Massimo Banzi προς τιμή του βασιλιά Arduin της Λιβρέας.Το πρώτο arduino κατασκευάστηκε στην Ιταλία το 2005 και είχε παρόμοια μορφή με αυτή που έχει και σήμερα (εικόνα 2.2):



Εικόνα 2.2:Το Πρώτο Arduino Που Κυκλοφόρησε

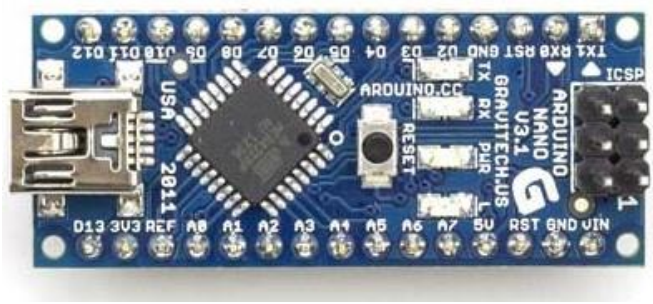
Τι είναι το arduino

Το arduino είναι μια πλακέτα ανοιχτού κώδικα με ενσωματωμένο μικροελεκτή καθώς και εισόδους/εξόδους η οποία προγραμματίζεται με μια γλώσσα αντικειμενοστραφούς προγραμματισμού την γλώσσα wiring μια γλώσσα παρόμοια με την C++[i1].Η πλατφόρμα arduino προσφέρεται για την δημιουργία εφαρμογών σε διαδραστικά αντικείμενα ή περιβάλοντα.Ουσιαστικά πρόκειται για ένα κύκλωμα όπου βασίζεται στον μικροελεγκτή ATmega της εταιρίας atmel.Εφόσον η πλακέτα είναι ανοιχτού κώδικα όλα τα σχέδια για την δημιουργία του υπάρχουν ελεύθερα στο διαδίκτυο έτσι ώστε ο καθένας να μπορεί να φτιάξει την δικιά του πλακέτα εάν το επιθυμεί.Έχοντας κατασκευάσει ή αγοράσει μια πλακέτα arduino μπορούμε να παρατηρήσουμε ότι συμπεριφέρεται σαν μικρός υπολογιστής ο οποίος είναι διαθέσιμος στο να συνδεθούν πάνω του ποικίλα εξαρτήματα εισόδου ή εξόδου και να μας φέρει το αποτέλεσμα που θέλουμε να πετύχουμε.

Το arduino σήμερα

Σήμερα το arduino είναι αναγνωρισμένο από πολλά πανεπιστημιακά ιδρύματα παγκοσμίως έχοντας πάνω από 200 προμηθευτές παγκοσμίως με το 90% των αγοραστών να βρίσκεται στην Ευρώπη και την Αμερική.Με το πέρασμα των χρόνων δημιουργήθηκαν και άλλες πιο εξελιγμένες εκδόσεις του arduino δύο από αυτές που ξεχώρισαν είναι το arduino mega και το arduino nano (εικόνες 2.3 και 2.4).

Εικόνα 2.3:Arduino Mega



Εικόνα 2.4:Arduino Nano

Ωστόσο το arduino επεκτάθηκε κλείνοντας συνεργασία με την εταιρία διεθνής φήμης Google.Η Google κυκλοφόρησε το android ADK(Accessory Development Kit) όπου βασίζεται στο arduino δίνοντας την δυνατότητα στους προγραμματιστές android να μπορούν να προγραμματίζουν ελέγχοντας τους αισθητήρες ενός κινητού τηλεφώνου όπως αισθητήρας κίνησης,οθόνη αφής και άλλα.Επίσεις λόγο του μεγάλου ενδιαφέροντος για το προιόν δημιουργήθηκαν και διάφορες ανεπίσημες εκδόσεις του όπου από αυτές ξεχώρισαν δύο από αυτές freeduino και seeduino (εικόνες 2.5 και 2.6).

Εικόνα 2.5:Freeduino

Εικόνα 2.6:Seeduino

Τα τεχνικά χαρακτηριστικά του arduino

Το arduino όπως είδαμε και στο προηγούμενο κεφάλαιο κυκλοφορεί επίσημα σε πολλές εκδόσεις όπου η κάθε έκδοση αποτελείτε από διαφορετικά χαρακτηριστικά η βασική έκδοση του είναι το arduino duemilanove η οποία αποτελεί μια σχετικά μεσαίων απαιτήσεων πλακέτα.

Η καρδιά του κάθε arduino είναι ο μικροελεγκτής, ο duemilanove αποτελείτε από τον ATmega328 όπου είναι ένας 8-bit μικροελεγκτής χρονισμένος στα 16MHz.Επίσης αποτελείτε από 3 ειδών μνήμες:

- Την μνήμη SRAM(static RAM) σε χωρητικότητα 2Kb όπου χρησιμεύει στο να μπορούν τα προγράμματα να αποθηκεύουν πινάκες,μεταβλητές κα. Όπως και σε μία ram τα δεδομένα σε πιθανή διακοπή ρεύματος χάνονται.
- Την μνήμη EEPROM(Electrically Erasable Programmable Read-Only Memory) στην χωρητικότητα του 1kb.Η μνήμη EEPROM σε αντίθεση με μια απλή ROM μπορεί να σβηστεί και να ξαναγραφεί πάρα πολλές φορές και χωρίς να υπάρχει απώλεια των δεδομένων χωρίς τροφοδοσία ρεύματος.Έτσι το arduino την χρησιμοποιεί για την αποθήκευση ή ανάγνωση δεδομένων χωρίς τύπο δεδομένων.
- Μια μνήμη Flash των 32kb.Τα 2kb από αυτά χρησιμοποιούνται από το firmware του arduino και είναι ήδη εγκατεστημένο από την αγορά του.Το firmware του arduino ονομάζεται bootloader και επιτρέπει στον προγραμματιστή να μπορεί να προγραμματίσει το arduino μέσω μιας απλής usb θύρας χωρίς να χρειάζεται κάποιο εξωτερικό hardware programmer.Τα υπόλοιπα 30kb χρησιμοποιούνται για την αποθήκευση προγραμμάτων που εγγράφονται από τον προγραμματιστή. Η μνήμη Flash όπως και η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset.Οι μνήμες flash υπό κανονικές συνθήκες δεν προορίζονται για να τρέχουν σε real time τα προγράμματα η συγκεκριμένη,λόγο του περιορισμένου χώρου, εμπεριέχει μια

βιβλιοθήκη που επιτρέπει να χρησιμοποιεί τον χώρο που περισσεύει από το πρόγραμμα για την real time εκτέλεση του[ι3].

Οι είσοδοι και οι έξοδοι

Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.Το arduino στην πάνω πλευρά έχει 14 θηλυκά pins που αριθμούνται από το 0 έως το 13 και μπορούν να χρησιμοποιηθούν ως ψηφιακές είσοδοι και έξοδοί.Τα pins λειτουργούν στα 5V επίσης μπορούν να δεχτούν η να παρέχουν έως και 40mA.Για να μπορέσουμε να προγραμματίσουμε το κάθε pin πότε θα δέχεται ρεύμα και πότε όχι μπορούμε να το κάνουμε με τις εντολές HIGH και LOW που θα δούμε πιο αναλυτικά παρακάτω.Ωστόσο τα 14 pins έχουν και άλλες λειτουργίες.

Το pin 0 και 1 λειτουργούν και σαν RX(receive) και TX(transmit) όταν στο πρόγραμμα είναι ενεργοποιημένη η σειριακή θύρα.Έτσι μόλις στέλνονται δεδομένα στην σειριακή μέσω usb αυτά προωθούνται και στην θύρα 1 οπού μπορούν να παρθούν από κάποια άλλη συσκευή όπως για παράδειγμα ένα άλλο arduino ή κάποιο πρόσθετο.

Τα pin 2 και 3 μπορούν να ρυθμιστούν μέσα από το πρόγραμμα του προγραμματιστή να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες εισόδους μόλις μεταβάλετε κάτι να σταματάει όλη η ροή του προγράμματος.Αυτά τα 2 pins μπορούμε να τ χρησιμοποιήσουμε σε projects που χρειάζονται μεγάλη ακρίβεια των αποτελεσμάτων.

Τα pins 3, 5, 6, 9, 10 και 11 χρησιμοποιούνται και ως ψευτοαναλογικές έξοδοι για το PWM (Pulse Width Modulation) που είναι το ίδιο σύστημα που χρησιμοποιούν και οι μητρικές κάρτες των Η/Υ για να μπορούν να ελέγχουν τους ανεμιστήρες ψύξης.Μπορεί να συνδεθεί για παράδειγμα σε αυτές τις θύρες ένα ανεμιστηράκι όπου θα ελέγχεται με ακρίβεια 8-bit(δηλαδή 256 καταστάσεις όπου για 0 θα είναι εντελώς σβηστό ενώ για 255 θα λειτουργεί σε πλήρη ισχύ).Όμως θα πρέπει να κατανοήσουμε ότι δεν λειτουργεί ακριβώς με αναλογικό σήμα αναλόγως με την τιμή που παίρνει στέλνει έναν παλμό μεταξύ από 0 έως 5V.

Στην κάτω πλευρά του arduino υπάρχουν άλλες 6 θύρες αριθμημένες από το 0 έως το 5 όπου λειτουργούν ως αναλογικές είσοδοι και μετατρέπουν το αναλογικό σήμα σε ψηφιακό.Επίσεις υπάρχει η δυνατότητα να μετατρέψουμε αυτές τις θύρες και να τις κάνουμε να λειτουργούν σαν τις απέναντι σε αυτήν την περίπτωση αυτές οι θύρες μετονομάζονται σε θύρες από 14 έως 19.

Τροφοδοσία arduino

Το arduino από την κατασκευή του μας δίνει δύο επιλογές στην τροφοδοσία του, μέσω της σύνδεσης usb με τον υπολογιστή ή μέσω ενός βύσματος των 2.1mm.Η τάση της τροφοδοσία μέσω του βύσμα πρέπει να είναι μεταξύ 7 και 12V.Αυτές τις απαιτήσεις μπορεί να ικανοποιήσει είτε έναν απλός μετασχηματιστής που να δίνει την τάση που θέλουμε ή μια μπαταρία ειδικά διαμορφωμένη για να ταιριάζει στην επαφή του βύσματος.Ωστόσο υπάρχουν άλλα έξι pins που σχετίζονται με την τροφοδοσία του arduino και το καθένα μπορεί να χρησιμοποιηθεί για διάφορες λειτουργίες.

Το πρώτο pin έχει την ένδειξη **RESET** που σημαίνει ότι όταν η συγκεκριμένη επαφή γειωθεί θα προκαλέσει αμέσως την επανεκκίνηση του arduino.

Το δεύτερο pin φέρει την ένδειξη **3.3V** και μπορεί να τροφοδοτήσει εξωτερικές συσκευές με 3.3V.Είναι σημαντικό να σημειώσουμε ότι η τάση που παρέχει δεν προέρχεται από την τροφοδοσία του arduino αλλά παράγεται από τον μικροελενκτή.

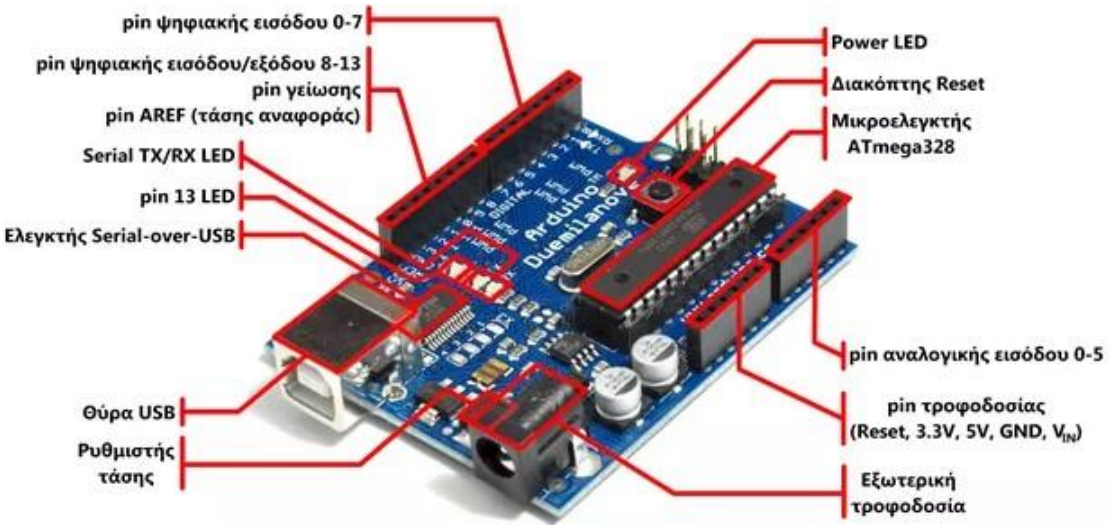
Το τρίτο pin έχει την ένδειξη **5V** και όπως και το δεύτερο pin τροφοδοτεί εξωτερικές συσκευές με 5V η διαφορά μεταξύ των δύο είναι ότι στο συγκεκριμένο pin η τάση προέρχεται από την τροφοδοσία της συσκευής απλά περνά από τον σταθεροποιητή τάσης για να έρθει στα 5V.

Τα επόμενα δύο έχουν ένδειξη **GND** και αφορούν την γείωση της συσκευής.

Το τελευταίο pin έχει την ένδειξη **Vin** και μπορεί να έχει διπλό ρόλο. Άμα την συνδυάσουμε με το διπλανό pin γείωσης μπορεί να τροφοδοτήσει εξολοκλήρου την συσκευή.Αν όμως η τροφοδοσία για την συσκευή είναι συνδεδεμένη μπορούμε να την χρησιμοποιήσουμε για παροχή τροφοδοσίας σε εξωτερικές συσκευές σε τάση από 7 έως 12V και χωρίς να περνάει από τον σταθεροποιητή τάσης όπως γίνεται με το τρίτο pin.

Ενσωματωμένο κουμπί και LEDs

Η πλακέτα του arduino εκτός από όσα περιγράψαμε παραπάνω διαθέτει και ένα κουμπί και 4 micro-LEDs.Το κουμπί χρησιμοποιείτε για να κάνει επανεκκίνηση της συσκευής.Το LED με την σήμανση power μόλις είναι αναμμένο μας δείχνει ότι η τροφοδοσία είναι συνδεδεμένη.Έτσι και στα LEDs με τις σημάνσεις RX και TX μας υποδεικνύουν πότε μεταφέρονται ή λαμβάνονται δεδομένα μέσω των συγκεκριμένων pins.Το τελευταίο LED με την ένδειξη L χρησιμοποιείτε σαν ένα απλό LED δηλαδή μπορούμε να του δώσουμε την κατάλληλη εντολή για να το κάνουμε να ανάβει ή να μένει σβηστό.Όσα περιγράψαμε παραπάνω φαίνονται στην παρακάτω εικόνα(εικόνα 2.7):

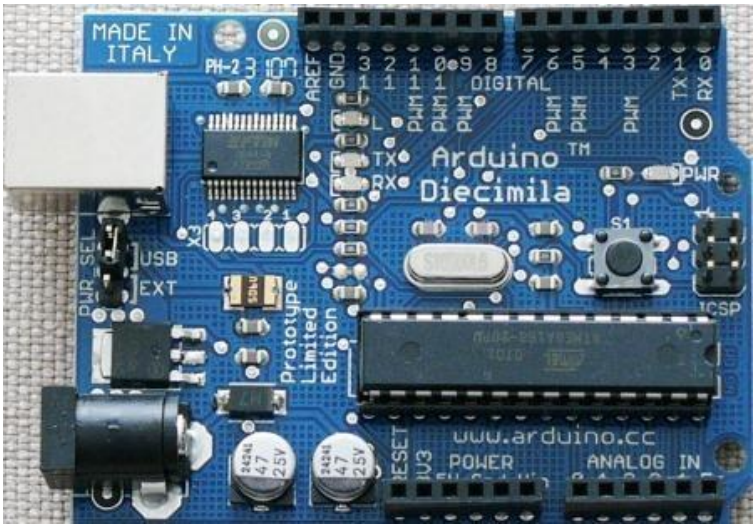


Εικόνα 2.7:Περιγραφή Arduino

Άλλες εκδόσεις του arduino

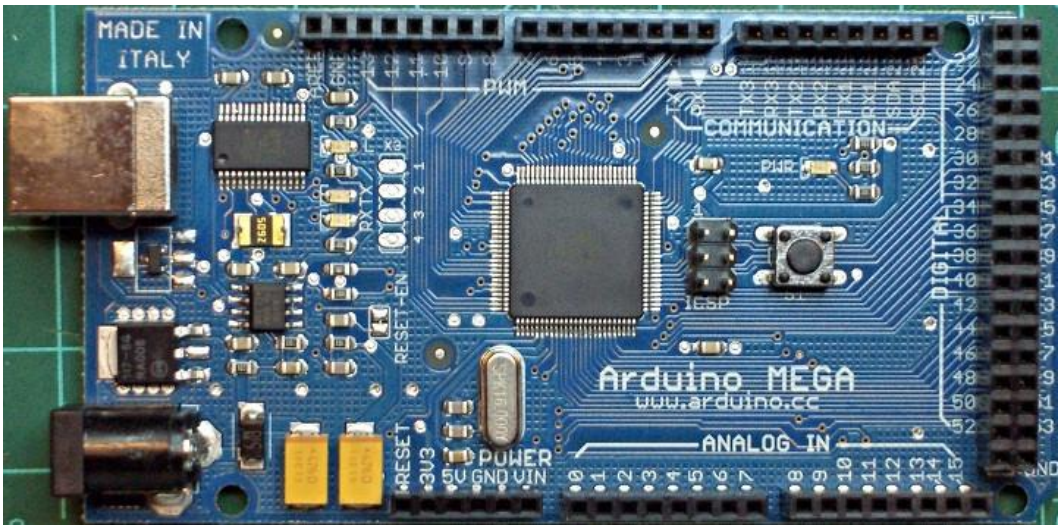
Έκτος από την έκδοση duemilanove έχουν ξεχωρίσει άλλες δύο εκδόσεις του arduino, αυτές είναι η έκδοση diecimila και οι έκδοση mega.Οι διαφορές με την βασική έκδοση είναι στα χαρακτηριστικά της κάθε πλακέτας.

Η έκδοση diecimila έχει μικροελενκτή με την μισή ακριβώς μνήμη από την βασική έκδοση. Διαθέτει τον ATmega168 με 1Kb SRAM, 512bytes EEPROM και 16Kb Flash.Είναι σημαντικό να σημειώσουμε ότι η συγκεκριμένη έκδοση έχει διακόπτη που καθορίζει από που θα τροφοδοτείται το arduino.Όσο αναφορά τις εισόδους/εξόδους είναι ακριβώς πανομοιότυπο με το duemilanove (εικόνα 2.8).



Εικόνα 2.8:Arduino Diecimila

Το arduino mega έχει τετραπλάσια μνήμη από την βασική έκδοση.Διαθέτει τον μικροελεγκτή ATmega1280 με 8Kb SRAM, 4Kb EEPROM και 128Kb Flash.Όμως τα πλεονεκτήματα δεν τελειώνουν μόνο στον μικροελεγκτή διαθέτει επίσης 40 επιπλέον ψηφιακές και 10 αναλογικές εισόδους/εξόδους.Επίσης έχει ακόμα 8 επαφές υποστήριξης ψευδοαναλογικής εξόδου PWM (εικόνα 2.9).



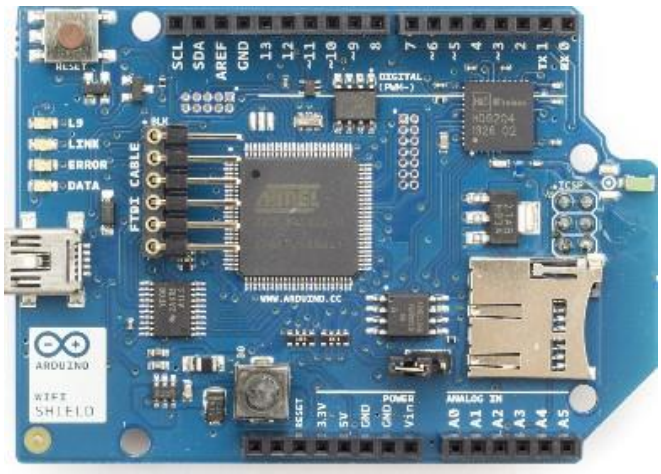
Εικόνα 2.9:Arduino Mega

Προκατασκευασμένα πρόσθετα για το arduino

Τα πρόσθετα για το arduino ονομάζονται shields.Τα shields είναι ολοκληρωμένες πλακέτες που είναι σχεδιασμένες ώστε να εφαρμόζουν πάνω στο Arduino προεκτείνοντας τις δυνατότητες του.Μερικά από τα πιο δημοφιλή πρόσθετα είναι:

- WiFi shield
- Διάφορα shield οθόνης
- GPS shield
- ProtoShield

Η WiFi shield παρέχει στο arduino δυνατότητες δικτύωσης μπορεί να συνδέεται σε ένα δίκτυο και να στέλνει ή να λαμβάνει πληροφορίες(εικόνα 2.10).



Εικόνα 2.10:WiFi Shield

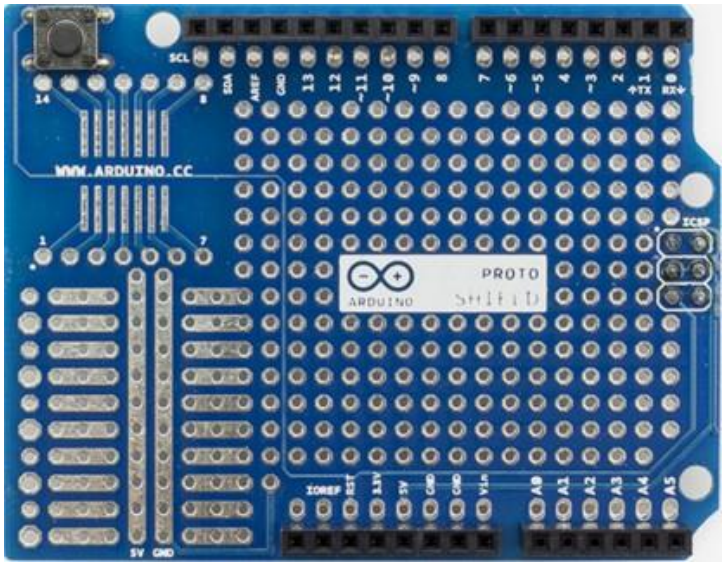
Έχουν κυκλοφορήσει πολλά shield για να προσθέτουν οθόνη στο arduino έτσι ώστε να προβάλει κάποια αποτελέσματα.Ένα shield οθόνης φαίνεται παρακάτω(εικόνα 2.11).

Εικόνα 2.11:LED Shield

Το gps shield δίνει την δυνατότητα στο arduino να στέλνει και να λαμβάνει στίγμα από gps.Το συγκεκριμένο shield χρειάζεται και μία κεραία για να λειτουργήσει (εικόνα 2.12).

Εικόνα 2.12:Gps Shield

Η proto shield είναι μια προσχεδιασμένη πλακέτα προτοτυποποίησης, συμβατή στις διαστάσεις του Arduino και χωρίς εξαρτήματα για την δημιουργία του δικού μας shield (εικόνα 2.13).



Εικόνα 2.13:Proto Shield

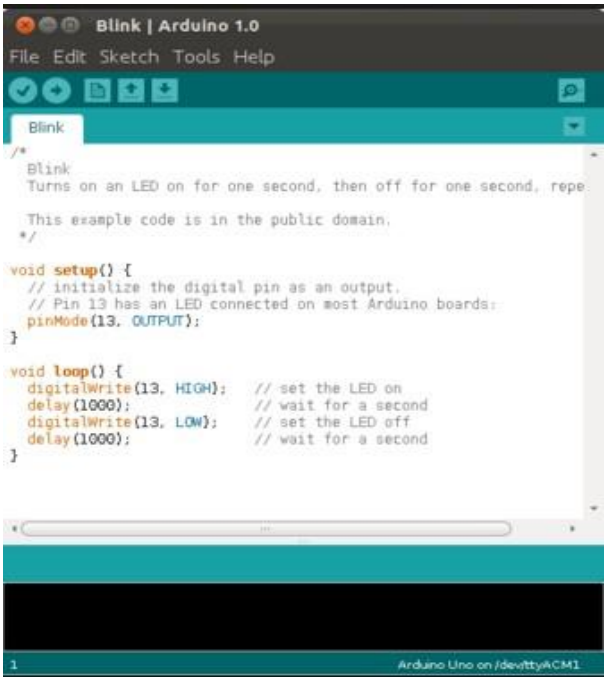
Οστώσο τα shield προωθούν τις υποδοχές του arduino έτσι ώστε να μπορούμε να συνδέσουμε πολλά shield μαζί ή και να συνδέσουμε τα shields σε συνδυασμό με δικά μας εξαρτήματα.

Το περιβάλλον προγραμματισμού arduino

Όπως και στην processing έτσι και το περιβάλλον του arduino είναι ένα πρόγραμμα ανοιχτού κώδικα και μπορούμε να το κατεβάσουμε δωρεάν από το επίσημο site του(www.arduino.cc).

Το πρόγραμμα μας δίνει την δυνατότητα να το κατεβάσουμε είτε για πλήρες εγκατάσταση στον υπολογιστή μας ή να το κατεβάσουμε για εκτέλεση χωρίς εγκατάσταση.

Το πρόγραμμα παρέχει στον προγραμματιστή έτοιμα παραδείγματα για την κατανόηση του arduino ,έναν compiler, μια οθόνη που ελέγχει την επικοινωνία με το arduino που έχουμε συνδέσει και την επιλογή sketch που ανεβάζει το πρόγραμμα που γράφουμε στο συνδεδεμένο arduino.Όπως παρατηρούμε στην παρακάτω εικόνα το περιβάλλον μοιάζει σε μεγάλο βαθμό με το περιβάλλον της processing επίσης παρατηρούμε ότι στο συγκεκριμένο πρόγραμμα είναι συνδεδεμένο ένα arduino uno και μπορούμε να το δούμε στην κάτω αριστερή γωνία (εικόνα 2.14).



Εικόνα 2.14:Προγραμματιστικό Περιβάλλον Arduino

Η wiring και η δομή του προγράμματος

Όπως αναφέραμε παραπάνω ο προγραμματισμός του arduino γίνεται με την γλώσσα προγραμματισμού wiring.Το πρόγραμμα αποτελείτε από τις βιβλιοθήκες και της μεταβλητές έπειτα πάντα από 2 βασικές συναρτήσεις όπου μέσα σε αυτές υπάρχουν η εντολές του προγράμματος στην συνέχεια ακολουθούν τα υπόλοιπες συναρτήσεις.Αυτές οι δύο συναρτήσεις είναι η **void setup()** (η ίδια όπως και στην processing) και η **void loop()**.Η χρήση αυτών των δύο συναρτήσεων είναι ίδια με των βασικών συναρτήσεων στην processing η πρώτη τρέχει με την εκκίνηση του προγράμματος και η δεύτερη τρέχει ακατάπαυστα.Παρακάτω φαίνεται η δομή του προγράμματος (εικόνα 2.15).

```
// Ενσωματώσεις βιβλιοθηκών, δηλώσεις μεταβλητών..

void setup()
{
  // ...
}

void loop()
{
  // ...
}

// Υπόλοιπες συναρτήσεις...
```

Εικόνα 2.5:Δομή Προγράμματος Σε Wiring

Λογικές μεταβλητές στην wiring

Στην γλώσσα προγραμματισμού wiring για τις μεταβλητές true ή false χρησιμοποιούνται από δύο διαφορετικές μεταβλητές.Για το true έχουμε το **INPUT** ή **HIGH** και για το false έχουμε το **OUTPUT** ή **LOW** και χρησιμοποιούνται αναλόγως με τα απαιτούμενα των συναρτήσεων που θα αναλύσουμε παρακάτω.

Βασικές εντολές για τον προγραμματισμό σε wiring

Η βασικότερη εντολή για τον καθορισμό λειτουργίας των εισόδων/εξόδων είναι η εντολή **pinMode(pin,mode)**.Η συγκεκριμένη εντολή καθορίζει το πότε ένα pin είναι ενεργό και μπαίνει στο πρώτο μέρος του προγράμματος στο setup.Το πρώτο όρισμα παίρνει το όνομα την επαφής και το δεύτερο παίρνει μια λογική τιμή(INPUT ή OUTPUT).

Εφόσον ενεργοποιήσουμε το pin για να το θέσουμε σε λειτουργία χρησιμοποιούμε την εντολή **digitalWrite(pin, mode)**.Με αυτήν την εντολή διοχετεύουμε ρεύμα στην επαφή.Το πρώτο όρισμα παίρνει το όνομα την επαφής και στο δεύτερο μια λογική τιμή(HIGH ή LOW).

Εάν στην εντολή pinmode έχουμε θέσει κατάσταση input χρησιμοποιούμε την εντολή **digitalRead(pin)**.Αυτή η εντολή παίρνει σαν όρισμα το όνομα της επαφής που είναι ορισμένη σε κατάσταση input και επιστρέφει μια λογική τιμή(0 για LOW ή 1 για HIGH).

Για τις αναλογικές θύρες υπάρχουν αντίστοιχα οι εντολές **analogRead(Pin)** και **analogWrite(pin, value)**.Η πρώτη εντολή παίρνει όρισμα την επαφή και επιστρέφει μια τιμή από το 0 έως το 1023 αναλόγως με τα volt που τροφοδοτείται.Η δεύτερη θέτει την θύρα σε ψευτοαναλογική λειτουργία, παίρνει ως όρισμα την επαφή και μια τιμή από το 0 έως το 255 αναλόγως με το σήμα που παράγεται.

Άλλη μια σημαντική εντολή είναι η εντολή **delay(ms)**.Αυτή η εντολή σταματά την ροή του προγράμματος για κάποια δευτερόλεπτα αναλόγως με τον αριθμό που έχουμε δώσει σαν όρισμα.

Το interrupt στον προγραμματισμό arduino

Το interrupt είναι μια εντολή όπου πυροδοτεί μία συνάρτηση έπειτα από κάποιο γεγονός.Στην wiring η εντολή είναι **attachInterrupt(interrupt,function,triggermode)** και έχει 3 ορίσματα.Το πρώτο όρισμα είναι το όνομα του συγκεκριμένου interrupt και ακολουθείται από το pin που επιρεάζεται,το δεύτερο όρισμα αναφέρεται στην συνάρτηση που πυροδοτείτε και το τελευταίο στο κάθε πότε θα ενεργοποιείται.Το τελευταίο όρισμα έχει 4 καταστάσεις που μπορεί να του δώσουμε **LOW,RISING,FALLING** και **CHANGE**.Το LOW ενεργοποιεί την συνάρτηση μόλις η κατάσταση του γίνει low.Το RISING πυροδοτείτε όταν από low γίνει high. Το FALLING πυροδοτείτε όταν από high γίνει low και το CHANGE όταν αλλάξει η κατάσταση στο pin που ενδιαφερόμαστε.Στο παρακάτω παράδειγμα αναφερόμαστε στο pin 13 και ενεργοποιεί την συνάρτηση blink οποιαδήποτε στιγμή αλλάξει η κατάσταση (εικόνα 2.16):

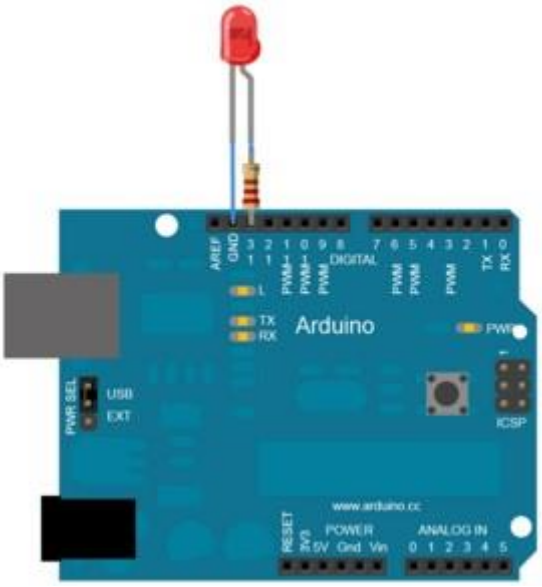
```
int pin = 13;
volatile int state = LOW;

void setup() {
    pinMode(pin, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(pin), blink, CHANGE);
}

void loop() {
    digitalWrite(pin, state);
}

void blink() {
    state = !state;
}
```

Εικόνα 2.16:Παράδειγμα attachInterrupt

Για να απενεργοποιήσουμε ένα interrupt χρησιμοποιούμε την εντολή **detachInterrupt(interrupt)**.Σαν όρισμα βάζουμε το όνομα του interrupt.Αν θέλουμε να σταματήσουμε όλα τα interrupts χρησιμοποιούμε την εντολή **noInterrupts()**.Εάν θέλουμε να επαναφέρουμε την λειτουργία τους μπορούμε να το κάνουμε με την εντολή **interrupts()**.Και στις δύο δεν χρησιμοποιούμε ορίσματα διότι αναφερόμαστε σε όλα τα interrupts.

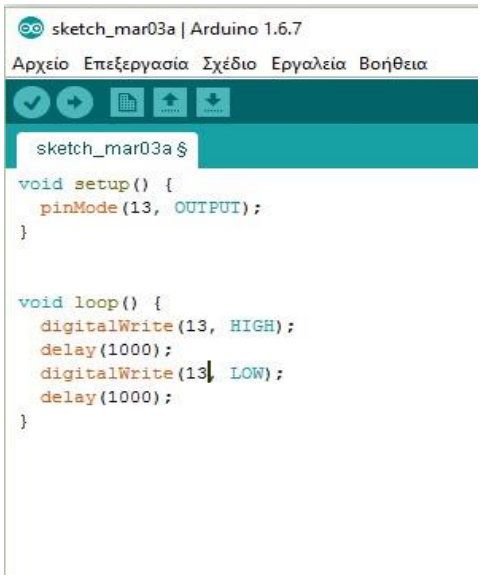
Λαμπάκι που αναβοσβήνει

Για να επιτύχουμε αυτό το παράδειγμα θα χρειαστούμε μια αντίσταση ohm, ένα λαμπάκι LED και την πλακέτα arduino.H συνδεσμολογία θα πρέπει να γίνει όπως φαίνεται στο παρακάτω σχήμα (εικόνα 2.17):



Εικόνα 2.17:Συνδεσμολογία Παραδείγματος

Ο κώδικας για να κάνει το LED να αναβοσβήνει θα περιέχει την αρχικοποίηση της διεπαφής την εντολή που θα ανάβει ο λαμπάκι,την εντολή που θα σβήνει το λαμπάκι και θα χωρίζονται μεταξύ τους με την εντολή delay.Ο κώδικας φαίνεται παρακάτω (εικόνα 2.18):



Εικόνα 2.18:Κώδικας Παραδείγματος

Κεφάλαιο 3

Περιγραφή Παιχνιδιού

Το παιχνίδι που δημιούργησα ονομάζεται "**I am superman**".Αποτελείτε από την αρχική οθόνη το κυρίως παιχνίδι και την τελική οθόνη που εμφανίζεται μόλις ο παίχτης χάσει όλες του τις ζωές.Την αρχική και την τελική οθόνη τις δημιούργησα στο Photoshop έτσι ώστε να εξυπηρετούν τις ανάγκες του παιχνιδιού.Το μεγαλύτερο σκορ που έχει καταφέρει ο παίχτης εμφανίζεται με την βοήθεια μιας συνάρτησης.Για να ξεκινήσει ο παίχτης ένα καινούριο παιχνίδι πρέπει να κάνει κλικ με το ποντίκι έπειτα ελέγχει την κλήση του χαρακτήρα με την απόσταση του χεριού από τον αισθητήρα.Η αρχική και η τελική οθόνη φαίνονται στις παρακάτω εικόνες (εικόνα 3.1 και 3.2):

Εικόνα 3.1:Αρχική Οθόνη

Εικόνα 3.2:Τελική Οθόνη

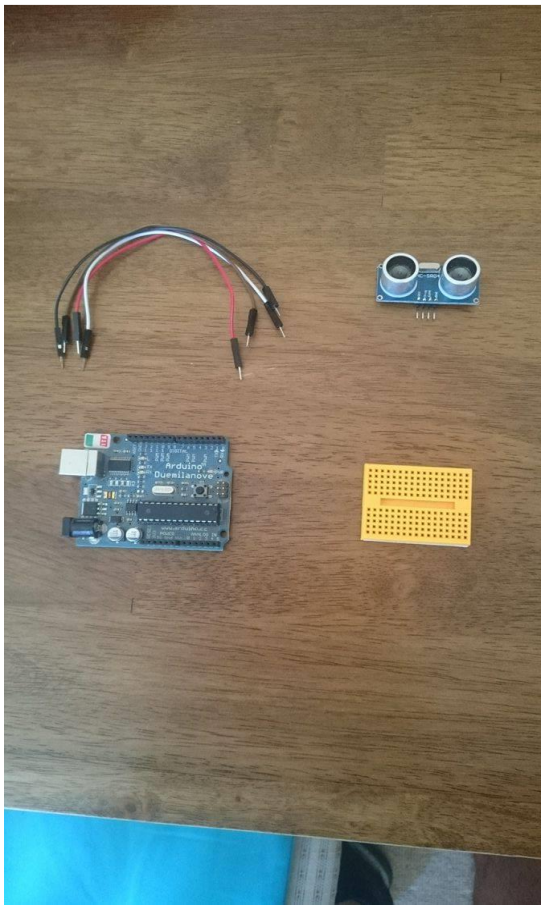
Ο κύριος σκοπός του παιχνιδιού είναι ο παίκτης να καταφέρει να μαζέψει όσο των δυνατών περισσότερα σύμβολα.Μόλις όμως δεν καταφέρει να πάρει ένα σύμβολο χάνει 1 ζωή. Οι ζωές του παίκτη και το συνολικό του σκορ εμφανίζονται στο κυρίως πρόγραμμα πάνω αριστερά και δεξιά αντίστοιχα.Ένα στιγμιότυπο από το κυρίως πρόγραμμα παρουσιάζεται παρακάτω (εικόνα 3.3) :

Εικόνα 3.3:Στιγμιότυπο Από Το Παιχνίδι

Απαραίτητα εργαλεία

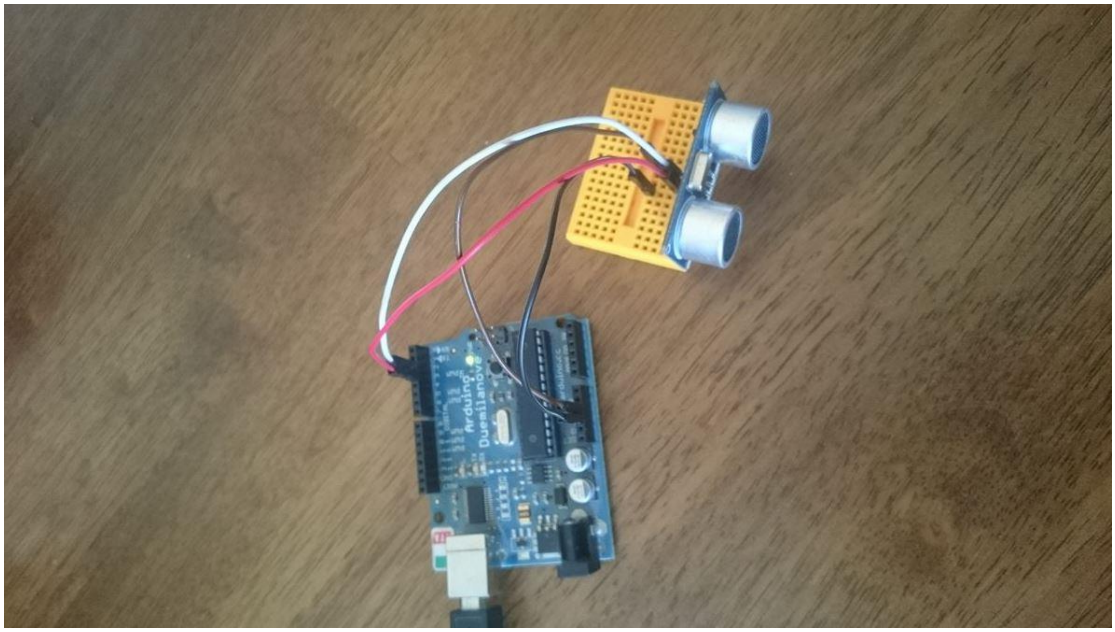
Για την δημιουργία αυτού του παιχνιδιού χρησιμοποίησα τα προγραμματιστικά περιβάλλοντα του arduino και της processing καθώς και το Photoshop για την επεξεργασία των εικόνων.Όσο αναφορά τα υλικά κομμάτια που χρησιμοποίησα είναι (εικόνα 3.4):

- Μια πλακέτα arduino (στο συγκεκριμένο την έκδοση duemilanove)
- Καλώδια
- Ένα breadboard
- Και τον υπερηχητικό αισθητήρα



Εικόνα 3.4:Υλικά Που Χρησιμοποιήθηκαν

Με την κατάλληλη συνδεσμολογία επιτεύχτηκε το επιθυμητό αποτέλεσμα. Η ολοκληρωμένη συνδεσμολογία φαίνεται στην παρακάτω εικόνα (εικόνα 3.5):



Εικόνα 3.5:Τελική Συνδεσμολογία

Βιβλιογραφία-Ηλεκτρονική Βιβλιογραφία

β: Getting started with Processing-Casey Reas and Ben Fry

ι1: www.wikipedia.com

ι2: www.processing.com

ι3: www.deltahacker.com

Παράρτημα

Κώδικας Arduino

```
int echoPin= 2;
int triggerPin= 3;
unsigned long pulsetime = 0;
unsigned distance =0;
unsigned OldDistance =0;

void setup (){
  pinMode (echoPin, INPUT);
  pinMode (triggerPin, OUTPUT);
  Serial.begin(9600);
}

void loop(){

  digitalWrite(triggerPin, LOW);
  delayMicroseconds(100);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(100);
  digitalWrite(triggerPin, LOW);
  //ipologisnos tis apostasis
  pulsetime = pulseIn(echoPin, HIGH);
  distance = pulsetime / 58;
  delay(10);
//alazei tin timi mono molis diaferei apo tin proigoumeni
if (OldDistance != distance) {

  Serial.println(distance);

  OldDistance = distance;
}

delay(50);
}
```

Κώδικας Processing

```
import processing.serial.*;

int i, j;
int Score ;
int DistanceUltra;
int IncomingDistance;
int gamestate=1;
int life;
int highscore=0;

float DistanceSuperIcon;
float ips;
float g;
float IconX;
float IconY;

PImage Icon;
PImage Superman;
PImage Start;
PImage GameOver;

Serial myPort;
String DataIn;
void setup()
{

  myPort = new Serial(this,"COM3", 9600);

    myPort.bufferUntil(10);

    frameRate(30);

    size(800, 600);
    rectMode(CORNERS) ;
    textSize(16);

    Icon = loadImage("icon.png");
    Superman = loadImage("superman.png");
    Start = loadImage("start.png");
    GameOver = loadImage("over.png");

}
```

```
46 void serialEvent(Serial p) {
47     DataIn = p.readString();
48 
49     IncomingDistance = int(trim(DataIn)); //metatrepei apo string se integer
50 
51 
52     if (IncomingDistance>1  && IncomingDistance<100 ) {
53         DistanceUltra = IncomingDistance;
54     }
55 }
56 
57 
58 void draw()
59 {
60   if(gamestate==1) {
61   image(Start,0,0);
62   }
63   if(gamesta PImage Start
64     background(0, 0, 0);
65     Ouranos();
66     fill(5, 72, 0);
69     //ipologismos tou score
70     DistanceSuperIcon = sqrt(pow((400-IconX), 2) + pow((ips-IconY), 2)) ;
71 
72     if (DistanceSuperIcon < 40) {
73         Score = Score+ 1;
74 
75         //dimiourgia neou stoxou
76         IconX = 900;
77         IconY = random(560);
78     }
79 
80     //life
81     textSize(60);
82     fill(0, 102, 153);
83     text("Lives :"+life,60,60);
84 
85 
86     //score
87     textSize(60);
88     fill(0, 102, 153);
89     text("Score :"+Score, 510, 60);
```

```
92      //g= mouseY-300;
93     g= (18- DistanceUltra)*4;
94
95     ips = ips + sin(radians(g))*10;
96
97      //kratаei ton superman stin othoni
98      if (ips < 0) {
99          ips=0;
100     }
101
102     if (ips > 600) {
103         ips=600;
104     }
105
106     TraceSuperman(ips, g);
107
108     IconX = IconX - cos(radians(g))*10;
109
110     if (IconX < -30) {
111       life=life-1;
112         IconX=900;
113         IconY = random(560);
114     }
116       image(Icon, IconX, IconY, 100, 60);
117
118    //life 0
119     if (life==0){
120    gamestate=2;
121    cursor();
122    if(Score>highscore)
123    {
124      highscore=Score;
125    }
126    }
127
128    //game over
129    if(gamestate==2){
130    image(GameOver,0,0);
131    //high score
132    textSize(60);
133    fill(20, 102, 133);
134    text("High Score:"+highscore,30,580);
135    }
136  }
```

```
}

void Ouranos() {
    noStroke();
    rectMode(CORNERS);

    for  (int i = 1; i < 600; i = i+10) {
        fill( 49    +i*0.165, 118   +i*0.118, 181  + i*0.075   );
        rect(0, i, 800, i+10);
    }
}


void TraceSuperman(float Y, float klisi) {

    noStroke();
    pushMatrix();
    translate(400, Y);
    rotate(radians(klisi));
    scale(0.5);
    image(Superman, -111, -55, 250, 150);
    popMatrix();
}
```

```
void mousePressed() {
  if(gamestate==1 || gamestate==2 ) {
    noCursor();
   gamestate = Score = 0;
    life=4;
    ips = 300; //arxiki timi tou ipsous tou superman
  }
}
```