Φύλλο Εργαστηριακής Άσκησης: **Ήχος (και ολίγο φως)**

Ονοματεπώνυμο:

Ημερομηνία: Ομάδα:

Ίσως η αφορμή για ένα ενδιαφέρον STEM PROJECT

Απλά επισημαίνουμε (εκ προοιμίου) πως μια κατασκευή με θέμα τη μουσική μπορεί να εξελιχθεί σε ένα εξαιρετικό STEM project.

Η σύνδεση με την Φυσική (ήχος, παραγωγή του, χαρακτηριστικά του, κλπ) και με τα Μαθηματικά (νότες, κλίμακες, αρμονία, κλπ) είναι δεδομένη.

Ίσως η σημερινή συνάντηση να δώσει ερεθίσματα σε κάποιες ομάδες να ασχοληθούν περισσότερο με το θέμα στην πορεία. Κάποια ελάχιστα ερεθίσματα έχουμε συμπεριλάβει στο σχετικό Παράρτημα του Φύλλου Εργασίας.

Γνωριμία με ένα νέο «εξάρτημα» - Ο βομβητής (buzzer/piezo/ηχειάκι)

Η πιο συνηθισμένη και δημοφιλής επιλογή για ηχητική επικοινωνία είναι το **buzzer**.
Ο **βομβητής** είναι συνηθισμένο στοιχείο στις ηλεκτρικές και ηλεκτρονικές συσκευές. Συχνά συναντάμε βομβητές σε συσκευές όπως ξυπνητήρια, κουδούνια πόρτας κ.ά.
Η κύρια λειτουργία αυτής της συσκευής είναι να μετατρέπει ένα ηλεκτρικό σήμα σε ήχο.

Σημαντική Σημείωση: Το σημερινό μας θέμα θα είναι απλό όσον αφορά στα κυκλώματα που θα δημιουργήσουμε.

Θα θέλαμε να δώσετε βάση και να κατανοήσετε τη δομή των προγραμμάτων που θα χρησιμοποιήσετε και θα δημιουργήσετε.

Θα εργαστούμε αποκλειστικά στο προγραμματιστικό περιβάλλον IDE (όχι στο S4A, στο οποίο η διαχείριση του θέματος «ήχος» γίνεται εξ ολοκλήρου και με πολύ «αξιοπρεπή» τρόπο με εντολές προς τα ηχεία του υπολογιστή μας).

Σημαντικό είναι (το ξαναλέμε) να δείτε αρκετά σημεία μέσα στα δείγματα κώδικα της σημερινής συνάντησης, γιατί είναι πιθανόν να τα χρειαστείτε (ως προγραμματιστικές τεχνικές) στις δικές σας ΤΕΛΙΚΕΣ κατασκευές.

Γνωριμία με το buzzer στην ΠΡΑΞΗ

ΠΡΩΤΟ ΒΗΜΑ: Συνδέουμε τους δυο ακροδέκτες του buzzer σε μια ψηφιακή έξοδο (π.χ. pin D8) και σε μια γείωση (GND) του Arduino.

Αυτό είναι το πρώτο απλό βασικό μας κύκλωμα.

Ακολουθεί το πιο απλό πρόγραμμα. Αντιγράψτε και επικολλήστε στο IDE. Ανεβάστε το στο Arduino.

`` void setup() { pinMode(8,OUTPUT); } void loop() { digitalWrite(8,HIGH); delay(200); digitalWrite(8,LOW); delay(200); } ```	Εδώ βλέπουμε την πιο απλή δομή προγράμματος ελέγχου του buzzer. Υποθέτουμε πως έχει συνδεθεί στο pin D8 και σε μια γείωση (GND). Αλλάξτε τις τιμές στα delay, πειραματιστείτε. Θυμηθείτε πως η τιμή που αναφέρουν είναι σε milliseconds.

ΔΟΚΙΜΕΣ

Παρατηρούμε ότι ο buzzer έχει δυο «ποδαράκια».

Αν διαβάσουμε κάποιες οδηγίες (θεωρία) θα δούμε πως οι βομβητές μπορούν να κατηγοριοποιηθούν σε δύο διαφορετικούς τύπους: ενεργών (active) βομβητών και παθητικών (passive) βομβητών, οι οποίοι έχουν και οι δύο πολικότητα (+ και -). Ας το έχετε γενικώς κατά νου ίσως μας χρειαστεί σε κάποια σημεία.

Αλλάξτε τη συνδεσμολογία για να δείτε αν η πολικότητα μας επηρεάζει (τουλάχιστον προς το παρόν).

ΔΕΥΤΕΡΟ ΒΗΜΑ: Προσθέστε ανάμεσα στο pin8 και στο «ποδαράκι του buzzer, μια φωτοαντίσταση. Δηλαδή παρεμβάλουμε μια μεταβλητή αντίσταση παραπάνω, άρα εύλογα υποθέτουμε πως στο buzzer «φτάνει» λιγότερο από 5V.

Με το ίδιο λογισμικό όπως και πριν δοκιμάστε τι συμβαίνει αν φωτίσουμε (με το φακό του κινητού) την φωτοαντίσταση και τι αν έχουμε φως δωματίου.

Μπορούμε να υποθέσουμε κάτι για την λειτουργία του buzzer;

Προαιρετικά πειραματιστείτε με μια άλλη μεταβλητή αντίσταση (ποτεσνιόμετρο) αντί για φωτοαντίσταση.

ΤΡΙΤΟ ΒΗΜΑ: Συνεχίζοντας την διερεύνηση ας συνδέσουμε το buzzer σε μια pwm (ψευδοαναλογική) έξοδο του Arduino, π.χ. την 9 (Θυμηθείτε, έχει μια ~ δίπλα στον αριθμό της)

Θα δοκιμάσουμε να δίνουμε διαφορετικές τιμές τάσης (όχι δηλαδή μόνο 5Volts όπως πριν. Για αυτό θα χρησιμοποιήσουμε την εντολή AnalogWrite()

`` int i; void setup() { pinMode(9,OUTPUT); } void loop() { for (i=0;i<=255;i=i+10) { analogWrite(9, i); delay(1000); } } ```	Εδώ είναι μια καλή ιδέα για να αλλάζουμε «αυτόματα» την τάση που στέλνουμε στο pin 9. Δείτε και κατανοήστε την λειτουργία της εντολής επανάληψης for. Λειτουργεί όσο η ακέραια (int) μεταβλητή i είναι από 0 έως 255 με βήμα αύξησης κάθε φορά 10. Αυτό επαναλαμβάνεται βέβαια καθώς περιλαμβάνεται μέσα στο void loop()

ΤΕΤΑΡΤΟ ΒΗΜΑ: Μπορείτε να κάνετε υποθέσεις για την λειτουργία του buzzer, σύμφωνα με τις ως τώρα παρατηρήσεις σας;

……

……

……

……

……

……

……

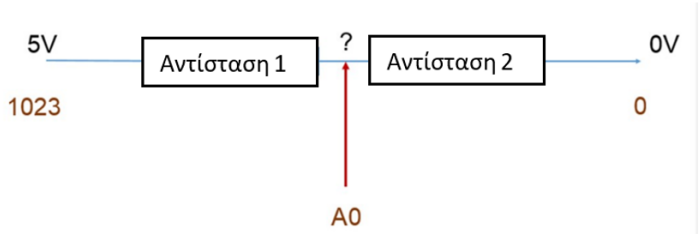
……

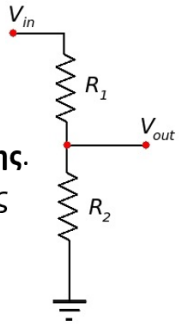
ΠΕΜΠΤΟ ΒΗΜΑ: ***Βρήκαμε κάπου έτοιμο το παρακάτω πρόγραμμα.***
Διαβάστε το, καταλάβετε τι κάνει (ή υποθέστε κάτι σχετικό) και χρησιμοποιήστε το.
Νομίζετε πως κάτι περισσεύει; (ας πούμε από τις μεταβλητές που έχουν δηλωθεί)
Όπως ελπίζουμε πως κατανοείτε το κύκλωμα είναι (όπως και στο ΠΡΩΤΟ ΒΗΜΑ) μόνο ένα buzzer ανάμεσα στο D8 και σε μια Γείωση.
Πειραματιστείτε λίγο (π.χ. με τα 80 και 100 στα for, και με τις τιμές στα delay).
Φτιάξτε μια δική σας μικρή «ακουστική ακολουθία» για να την χρησιμοποιήσετε αργότερα.

```
int buzzer=8;      //set the digital IO pin to control the buzzer
void setup()
{
pinMode(buzzer,OUTPUT);    //set digital IO pin as output mode
}
void loop()
{
unsigned char i,j;    //Define variable
while(1)
{
for(i=0;i<80;i++)     //Make sound with same frequency
{
digitalWrite(buzzer,HIGH);     //Make sound
delay(1);                      //Delay 1ms
digitalWrite(buzzer,LOW);    //No sound
delay(1);                     //Delay 1ms
}
for(i=0;i<100;i++)            // Make sound with other frequency
{
digitalWrite(buzzer,HIGH);     //Make Sound
delay(2);                      //Delay2ms
digitalWrite(buzzer,LOW);     //No sounds
delay(2);                      //Delay 2ms
}
}
}
```

Σύνδεση με παλιότερες εφαρμογές μας

Θυμηθείτε τι είχαμε δει για ένα διαιρέτη τάσης με φωτοαντίσταση



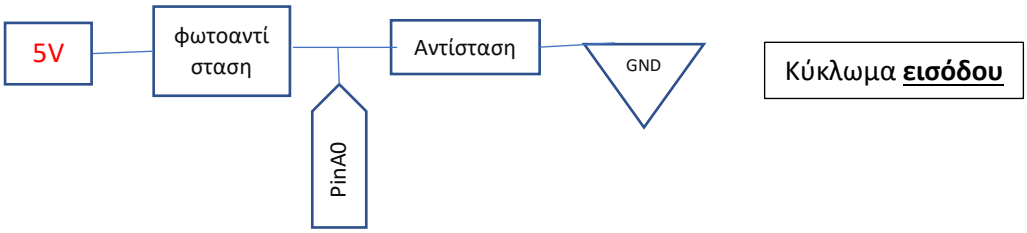


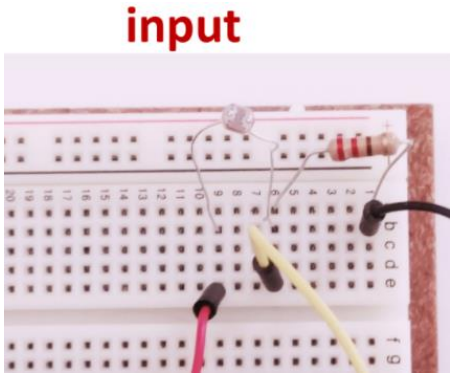
- Κυκλώματα όπως το παραπάνω, το οποίο αποτελείται από δύο αντιστάτες συνδεδεμένους σε σειρά, στα άκρα των οποίων εφαρμόζεται η τάση εισόδου (π.χ. 5V), ονομάζονται **διαιρέτες τάσης**.
- Τάση εξόδου είναι η διαφορά δυναμικού ανάμεσα στους ακροδέκτες της μίας από τις δύο αντιστάσεις. Οι τιμές που μπορεί να πάρει η τάση εξόδου κυμαίνονται από το 0 έως την τάση εισόδου.

Είχαμε χρησιμοποιήσει αυτή τη διάταξη με μια αντίσταση και μια φωτοαντίσταση ώστε ΑΝΑΛΟΓΑ με την τιμή σε μια αναλογική είσοδο (π.χ. την A0) να ενεργοποιούμε ή όχι ένα φως (led).

Τώρα θέλουμε να φτιάξουμε ένα παρόμοιο μηχανισμό, μόνο που τώρα θα ενεργοποιείται ένα ηχητικό σήμα όταν πλησιάσει ένα φως στο Arduino μας.

Θυμίζουμε ένα πρόχειρο σχεδίασμα κυκλώματος Εισόδου





Το Κύκλωμα Εξόδου είναι απλά ένα buzzer ανάμεσα στο pin D8 (ας πούμε) και σε μια γείωση.

Φτιάξτε τώρα ένα κατάλληλο πρόγραμμα.

Αν θέλετε μπορείτε να συμβουλευτείτε το ακόλουθο πρόγραμμα.

Ο ήχος που παράγεται είναι στηριγμένος στον κώδικα του ΠΕΜΠΤΟΥ ΒΗΜΑΤΟΣ μετά από μικροπειραματισμούς, η λογική του δομή όμως, στηρίζεται στη λογική του διαιρέτη τάσης που έχουμε ξαναδεί στο παρελθόν. Η τιμή 100 στην εντολή if, βγαίνει μετά από πειραματισμό και φυσικά σχετίζεται με την τιμή της αντίστασης που χρησιμοποιήσαμε (π.χ. 220ΚΩ).

```
int buzzer=8;
int zzz;

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

void loop()
{
unsigned char i;

zzz=analogRead(A0);
if (zzz>100)
   {
     for(i=0;i<100;i++)
       {
        digitalWrite(buzzer,HIGH);
        delay(3);
        digitalWrite(buzzer,LOW);
        delay(3);
       }
    for(i=0;i<50;i++)
       {
        digitalWrite(buzzer,HIGH);
        delay(10);
        digitalWrite(buzzer,LOW);
        delay(10);
       }
   }
}
```

Η ΠΡΩΤΗ μας ΔΙΕΡΕΥΝΗΣΗ ολοκληρώθηκε.
Στο δεύτερο μέρος θα δούμε κάτι λίγο πιο «μελωδικό»

ΔΕΥΤΕΡΟ ΜΕΡΟΣ _ ΜΟΥΣΙΚΗ?

ΠΡΩΤΟ ΒΗΜΑ: Βρήκαμε το παρακάτω πρόγραμμα. Φορτώστε το στο IDE και δείτε τι κάνει.

```
#define NOTE_C3  131
#define NOTE_D3  147
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
#define NOTE_C4  262

#define BUZZER_PIN 8

int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {

  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);

    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

Όπως παρατηρούμε παίζει μια απλή μελωδία ΜΙΑ φορά.

ΔΕΥΤΕΡΟ ΒΗΜΑ: Κάντε κάποιες δοκιμές (ακόμη και αν δεν καταλαβαίνεται πολύ καλά τι κάνει κάθε μια εντολή) ώστε να επαναλαμβάνεται η ίδια μελωδία συνεχώς.
Αν δεν τα καταφέρετε δείτε την ακόλουθη λύση. Εντοπίζετε τις διαφορές;

```
#define NOTE_C3  131
#define NOTE_D3  147
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
#define NOTE_C4  262

#define BUZZER_PIN 8

int melody[] = {

  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {

}

void loop() {

  // iterate over the notes of the melody:

  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);

    // stop the tone playing:

    noTone(8);
     }
  delay (2000);
}
```

ΤΡΙΤΟ ΒΗΜΑ: ΧΩΡΙΣ ΚΑΤΑΣΚΕΥΗ!!!

Κάτι ακόμα στις Προγραμματιστικές τεχνικές. Τι είναι μια υπορρουτίνα; (μια συνάρτηση;))

Χωρίς να μεταφέρετε τον παρακάτω κώδικα στο IDE απλά παρατηρήστε την δομή του.
Το πρόγραμμα είναι μισοφτιαγμένο και έχει σκοπό να είναι μια βάση για πειραματισμούς.

Παρατηρήστε ότι χρησιμοποιεί την «εντολή» **play_note()**
Αυτή δεν είναι μια εντολή που την γνωρίζει το IDE. Την φτιάξαμε εμείς (!!!) γιατί την χρειαστήκαμε. Δηλαδή θα θέλαμε κάτι τέτοιο γιατί το χρησιμοποιούμε πολλές φορές.

Δείτε στο κάτω μέρος του πηγαίου κώδικα τον τρόπο που δηλώνουμε τι κάνει αυτή η νέα εντολή (συνάρτηση).

```
int NOTE_C4 = 262;
// οι αριθμοί αντιστοιχουν σε συχνοτητες σε Hz
int NOTE_D4 = 294;
int NOTE_E4 = 330;
int NOTE_F4 = 349;
int NOTE_G4 = 392;
int NOTE_A4 = 440;
int NOTE_B4 = 494;
int NOTE_C5 = 523;
void setup() {
 pinMode (9, OUTPUT);
}
void loop()
{  play_note (NOTE_C4);
  delay (300);
//πειραματιστειτε με τις παυσεις...
  play_note (NOTE_D4);
  delay (300);
 //συνεχίστε μόνοι σας την μελωδία...
}

void play_note (int nota)
{
  for (int i = 0; i <= 200; i++)
 // πειραματειστειτε με τον χρονο
  {
  digitalWrite (9, HIGH);
  delay (1000/nota); //σε τι αντιστοιχεί η διαίρεση;
  digitalWrite (9, LOW);
  delay (1000/nota);
  }
}
```

ΤΕΤΑΡΤΟ ΒΗΜΑ: Παίζουμε το μουσικό θέμα από τη σειρά ταινιών «Ροζ Πάνθηρας»

Φορτώστε το αντίστοιχο πρόγραμμα στο IDE (Παράρτημα 1). Ακούστε τη μελωδία.

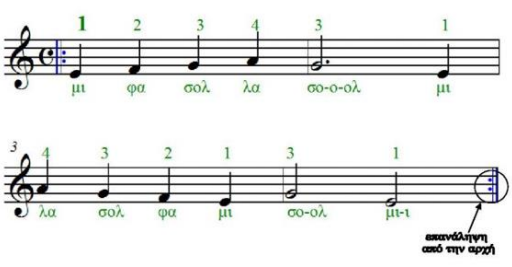
ΠΕΜΠΤΟ ΒΗΜΑ: Κάποιες παρατηρήσεις σε αυτόν τον κώδικα.

- Η λογική που «φτιάχνουμε μουσική» μένει να διερευνηθεί από εσάς με βάση τα σχόλια που περιέχονται στον κώδικα. Λίγες μουσικές γνώσεις είναι αλήθεια πως θα βοηθούσαν, γι αυτό και το θεωρούμε σε αυτή τη φάση μη απαραίτητο.
- Παρατηρούμε την εντολή **tone(BUZZER_PIN, melody[note], duration);**
  Με την εντολή tone() δίνουμε οδηγία για ήχο με τρεις παραμέτρους.
  Το pin που θα δώσουμε ήχο, την «νότα», και την διάρκεια του ήχου.
- Υπάρχουν και χρησιμοποιούνται δυο **Πίνακες**
  melody[] = { ……..} και durations[] = {……..}
  Ο πρώτος περιλαμβάνει μια λίστα/σειρά από «νότες» και ο δεύτερος την αντίστοιχη σειρά από την διάρκεια της κάθε νότας.
  Αυτά τα ζευγάρια καλούνται διαδοχικά από την εντολή tone() και έτσι έχουμε το μουσικό αποτέλεσμα.
  Ο μηχανισμός αυτός πιστεύουμε πως μπορεί να είναι «ορατός» από εσάς μέσα στον κώδικα.
- Υπάρχει μια οδηγία **#include "pitches.h"** που είναι απενεργοποιημένη με τα δυο // μπροστά της που την κάνουν σχόλιο.
  Στην πραγματικότητα θα ‘πρεπε να προσθέσουμε αυτή την «βιβλιοθήκη» στο IDE, αλλά αυτό κάπου δεν δούλεψε.
  Αυτή περιείχε τις συχνότητες για κάθε νότα, μια σειρά από δηλώσεις του τύπου **#define NOTE_B0 31** (δηλαδή σε αυτό το παράδειγμα δίνει στην νότα B0 την συχνότητά της (31).
  Έτσι συμπεριλάβαμε στον κώδικα στη αρχή, ως δηλώσεις σταθερών (constants) μια μεγάλη λίστα που αποδίδει τις κατάλληλες συχνότητες σε όσες νότες θα μπορούσε να «παίξει» ένα buzzer.

Με την ευκαιρία, θα μπορούσατε να ψάξετε (ή να ρωτήσετε κάποιον λίγο σχετικό) τη σχέση που έχουν οι νότες με τις συχνότητες. Υπάρχουν κανόνες και με τη γνώση της συχνότητας μιας μόνο βασικής νότας (Α-Λα 440Hz) μπορούμε να έχουμε τις συχνότητες για όλες τις νότες (με πολύ απλούς υπολογισμούς). Δείτε κάποιες εικόνες στο Παράρτημα 2

ΤΕΛΕΥΤΑΙΟ ΒΗΜΑ: Μια προαιρετική πρόκληση

Με την πληροφορία που μπορείτε να βγάλετε από την εικόνα, κάντε το buzzer του Arduino σας να κελαηδήσει αυτόν τον ρυθμό. Χρειάζεται, είναι η αλήθεια, λίγες γνώσεις μουσικής.



Υπόδειξη:

Μετατρέψτε τις νότες από Λα-Σι-Ντο-… σε A-B-C-… κλπ

Η διάρκεια κάθε νότας φαίνεται και παραστατικά κάτω από το πεντάγραμμο, πέρα από τη μουσική σημειογραφία.

«Ταΐστε» με αυτά τα δεδομένα τους δυο πίνακες και εκμεταλλευτείτε την προγραμματιστική δομή και τις τεχνικές του «Ροζ Πάνθηρα»

ΠΑΡΑΡΤΗΜΑ 1 ΡΟΖ ΠΑΝΘΗΡΑΣ

```
/*
************************************************
 Public Constants
 ************************************************
 */

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
```

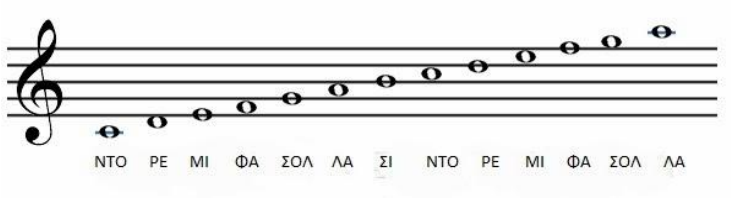
```
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978

/*

 Melody
 Plays a melody
 circuit:
 - 8 ohm speaker on digital pin 8

*/

// #include "pitches.h"


#define BUZZER_PIN 8
#define REST 0

int melody[] = {
 REST, REST, REST, NOTE_DS4,
 NOTE_E4, REST, NOTE_FS4, NOTE_G4, REST, NOTE_DS4,
 NOTE_E4, NOTE_FS4,  NOTE_G4, NOTE_C5, NOTE_B4, NOTE_E4, NOTE_G4, NOTE_B4,
 NOTE_AS4, NOTE_A4, NOTE_G4, NOTE_E4, NOTE_D4,
 NOTE_E4, REST, REST, NOTE_DS4,

 NOTE_E4, REST, NOTE_FS4, NOTE_G4, REST, NOTE_DS4,
 NOTE_E4, NOTE_FS4,  NOTE_G4, NOTE_C5, NOTE_B4, NOTE_G4, NOTE_B4, NOTE_E5,
 NOTE_DS5,
 NOTE_D5, REST, REST, NOTE_DS4,
 NOTE_E4, REST, NOTE_FS4, NOTE_G4, REST, NOTE_DS4,
 NOTE_E4, NOTE_FS4,  NOTE_G4, NOTE_C5, NOTE_B4, NOTE_E4, NOTE_G4, NOTE_B4,
```

```
  NOTE_AS4, NOTE_A4, NOTE_G4, NOTE_E4, NOTE_D4,
  NOTE_E4, REST,
  REST, NOTE_E5, NOTE_D5, NOTE_B4, NOTE_A4, NOTE_G4, NOTE_E4,
  NOTE_AS4, NOTE_A4, NOTE_AS4, NOTE_A4, NOTE_AS4, NOTE_A4, NOTE_AS4, NOTE_A4,
  NOTE_G4, NOTE_E4, NOTE_D4, NOTE_E4, NOTE_E4, NOTE_E4
};

int durations[] = {
  2, 4, 8, 8,
  4, 8, 8, 4, 8, 8,
  8, 8,  8, 8, 8, 8, 8, 8,
  2, 16, 16, 16, 16,
  2, 4, 8, 4,

  4, 8, 8, 4, 8, 8,
  8, 8,  8, 8, 8, 8, 8, 8,
  1,
  2, 4, 8, 8,
  4, 8, 8, 4, 8, 8,
  8, 8,  8, 8, 8, 8, 8, 8,

  2, 16, 16, 16, 16,
  4, 4,
  4, 8, 8, 8, 8, 8, 8,
  16, 8, 16, 8, 16, 8, 16, 8,
  16, 16, 16, 16, 16, 2
};

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

void loop()
{
  int size = sizeof(durations) / sizeof(int);

  for (int note = 0; note < size; note++) {
    //to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int duration = 1000 / durations[note];
    tone(BUZZER_PIN, melody[note], duration);

    //to distinguish the notes, set a minimum time between them.
    //the note's duration + 30% seems to work well:
    int pauseBetweenNotes = duration * 1.30;
    delay(pauseBetweenNotes);

    //stop the tone playing:
    noTone(BUZZER_PIN);
  }
}
```

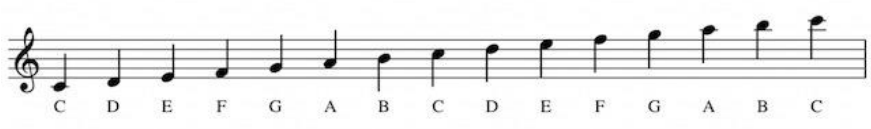
ΠΑΡΑΡΤΗΜΑ 2: Για ένα STEM PROJECT «ΜΟΥΣΙΚΗ»

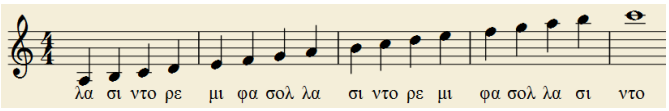
Απλά επισημαίνουμε ξανά πως μια κατασκευή με θέμα τη μουσική μπορεί να εξελιχθεί σε ένα εξαιρετικό STEM project. Η σύνδεση με την Φυσική (ήχος, παραγωγή του, χαρακτηριστικά του, κλπ) και με τα Μαθηματικά (νότες, κλίμακες, αρμονία, κλπ) είναι δεδομένη.

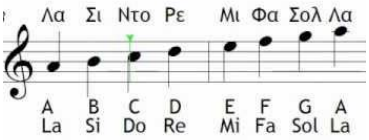
Δείτε κάποιες χαρακτηριστικές εικόνες σαν ερέθισμα για περαιτέρω αναζητήσεις.

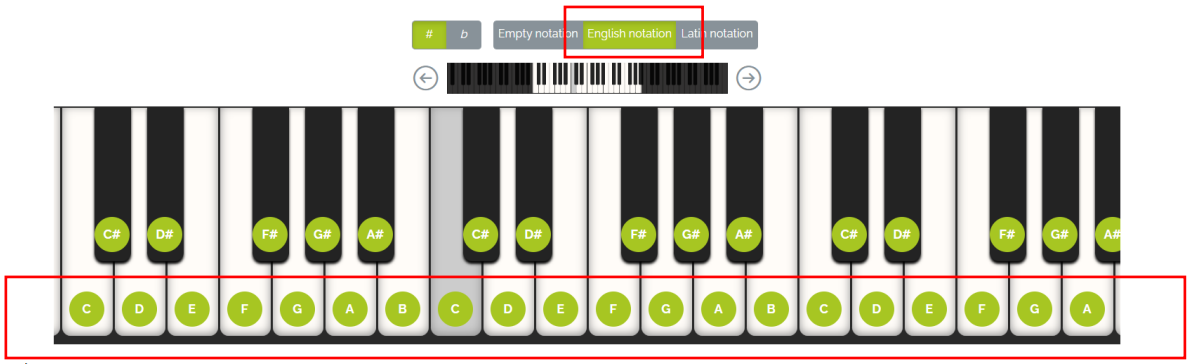


Οι νότες στο κλειδί του ΣΟΛ

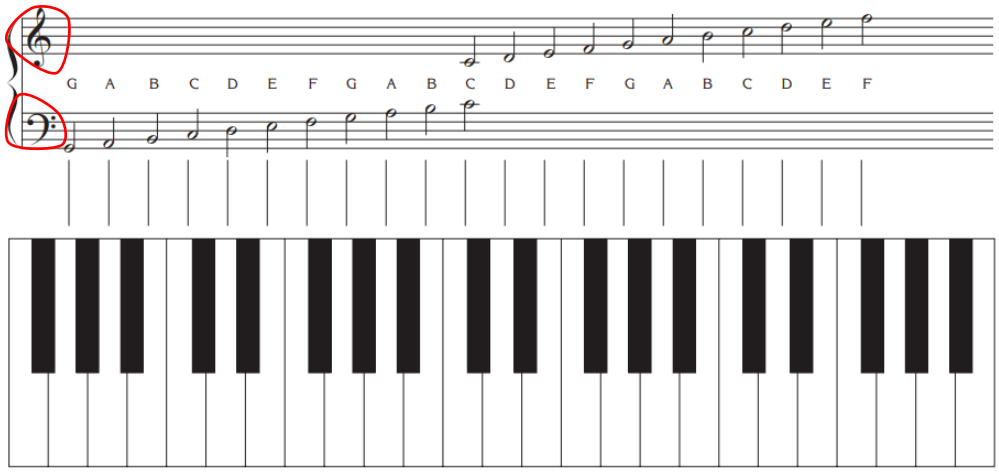








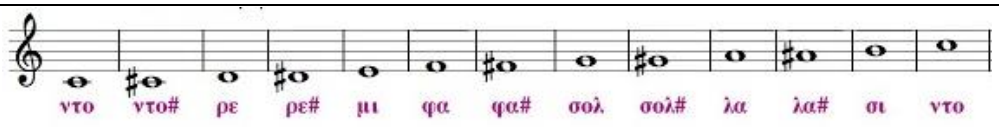
Οι δυο παραπάνω (αρχικές) εικόνες από το
https://www.imusic-school.com/en/tools/online-piano-keyboard/



Ο πρώτος (θεμελιώδης) και ο δεύτερος αρμονικός έχουν σχέση συχνοτήτων **f2 = 2f1**
και δίνουν λόγο **2:1** (π.χ. από ΛΑ σε ΛΑ)

$f_{\Lambda\alpha 4} = 440Hz$

$f_{\Lambda\alpha 5} = 880Hz$



ΧΡΩΜΑΤΙΚΗ ΚΛΙΜΑΚΑ

Η χρωματική κλίμακα περιλαμβάνει και τα ημιτόνια (υφέσεις/διέσεις).

Ανάμεσα π.χ. σε δυο ΝΤΟ (που έχουν λόγο συχνοτήτων 1 προς 2), υπάρχουν [στη σύγχρονη μουσική] 12 διαστήματα, που κατανέμονται στις ενδιάμεσες συχνότητες. (αντίστοιχα ανάμεσα σε δυο ΛΑ κλπ)

$f_{\Lambda\alpha 4} = 440Hz$

$f_{\Lambda\alpha 5} = 880Hz$

NOTE	FREQUENCY f, Hz	WAVE LENGTH l, m	RATIO l/lC	RATIO l/lC
C	261.63	1.3	1.0	3.0
C#	277.18	1.23	1.059	3.2
D	293.66	1.16	1.122	3.4
D#	311.13	1.09	1.189	3.6
E	329.63	1.03	1.26	3.8
F	349.23	0.97	1.335	4.0
F#	369.99	0.92	1.414	4.2
G	392	0.87	1.498	4.5
G#	415.30	0.82	1.587	4.8
A	440	0.77	1.682	5.0
A#	466.16	0.73	1.782	5.3
B	493.88	0.69	1.888	5.7
C	523.25	0.65	2.0	6.0

C=Ντο A=Λα