Curs 6

Dec. 2019

Funcții Labview pentru prelucrarea sunetelor

Prof.dr.ing. Iulian Lupea, UTCluj

1. Noțiuni despre placa de sunet PS

O placă de sunet/ placă audio (**sound card/audio card**) este o componentă a calculatorului condusă de programe, permițând manipularea semnalelor audio şi intrări/ieşiri de semnale audio.

Placa audio poate fi:

*integrată în placa de bază sau

*o extensie/placă conectată prin PCI, ISA, USB, PCMCIA, PCExpress etc.

Placa de Sunet are conectori/ porturi:

1. „**line in**” Light blue pentru semnal provenit de la casetofon (cassette tape recorder). Semnalul este digitizat şi memorat pe HDisk şi eventual procesare ulterioară.

2. **intrare pentru microfon** extern Pink printr-un (microphone jack) folosit pentru înregistrare mesaj sonor şi eventual recunoaşterea vorbirii.
3. **ieşire pentru căşti** (headphone jack) Lime green pentru ascultare de MP3s, DVDs sau alte fişiere audio în mod privat.

Calitatea difuzoarelor este dată de:

Frequency response, Total Harmonic Distortion (THD) şi Watts.

4. **conector pentru interfața MIDI** /digital **Musical Instrument Digital Interface** Gold/Grey este standard pentru reprezentare şi transmitere de sunete digitale.
5. **ieşire pentru difuzoare spate** stereo Black

PS poate manipula fişiere .wav, .mp3 şi .cda sau alte formate.

PlacaSunet are drivere şi soft propriu.

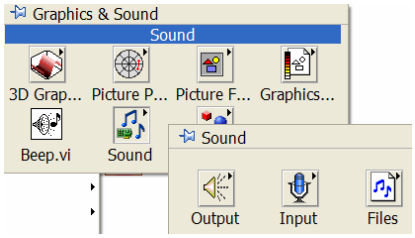
Placa de Sunet conține 2 **convertoare**:

• unul **ADC** (analog to digital) = conversie semnal analogic de la microfon în semnal digital pentru prelucrare ulterioară în calculator şi

• unul **DAC** digital-to-analog care converteşte semnalul din format digital în format analogic pentru alimentarea unui amplificator, căşti etc. (prin intermediul unui conector TRS sau RCA).

Plăci de sunet	Frecvența de eşantionare (sampling rate)	Rezoluția biți per eşantion
comune	44.1 kHz sau 48 kHz	16
performante	96 kHz sau chiar 192 kHz	24

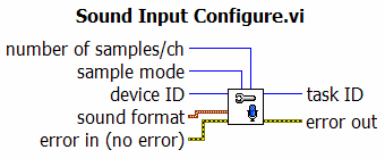
Paleta “Sound” din Labview conține funcții pentru gestionarea plăcii de sunet →



2. Funcții pentru *ACHIZIȚIE* de sunet prin PlacaSunet (convertorul ADC)

2.1. Lanțul funcțiilor de bază pentru achiziție de sunet:

› **SI Config.vi** : pregăteşte placa de sunet pentru achiziţie, alocă **buffer RAM** de memorie intermediar ex: 8192 bytes, apoi trimite datele in buffer intermediar de memorie.



number of samples/ch specifică numărul de *eşantioane/ pe canal* care se alocă în buffer.

sample mode=

Finite Samples (VI achizitionează numărul de eşantioane specificat şi se opreşte)

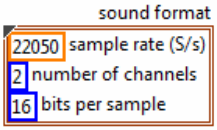
Continuous Samples = achiziţie continuă (funcţia SI Read.vi se va apela repetitiv).

device ID=0 implicit → număr dispozitiv achiziţie (placa sunet)

sound format=propune **format sunet** (structură de date):

1) rata eşantionare: 44100 eş/sec (Hz),
22050 (implicit),
11025,

2) nr. canale de achiziţie:
1(mono) /
2(stereo),



3) 8/16 biţi/eşantion
- biţi necesari pentru memorarea unui eşantion (16=implicit).

Obs:

11.025 kHz şi **22.05 kHz** sunt frecvenţe de eşantionare folosite în fişiere WAV.

44.1 kHz sampling rate şi 16 bits per sample sunt comune pentru compact disc digital audio (CD-DA). Frecvenţa de eşantionare de 44.1kHz este folosită şi de formatul MP3 (standard Sony).

88.2 kHz şi **176.4 kHz** frecvenţe de eşantionare înalte folosite in DVD-Audio

Funcţia **returnează:**

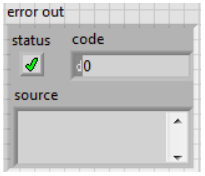
task ID= generează *număr de identificare task*

error out = conţine un cluster cu 3 câmpuri:

status *(tip boolean)* = TRUE dacă a apărut o eroare, False altfel,

code (tip numeric intreg)= codul erorii apărute dacă status = true,

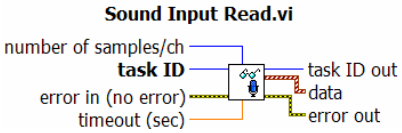
source (şir de caractere)=specifică sursa erorii sau atenţionării = şirul reprezintă numele nodului care a generat eroarea.



› **SI Read.vi**: transferă/citeşte datele din buffer RAM în memoria calcul. tablou de waveforms;

Intrări:

number of samples/ch: specifică numărul de eşantioane/pe channel de citit din buffer, la o citire



Returnează:

data = array of waveforms

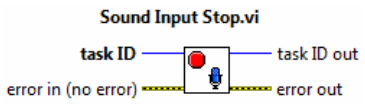
fiecare waveform=semnal de pe un canal, conţine:

t0=timp start,

dt=1/ rata eşantionare (de exemplu Fs=22050 Hz),

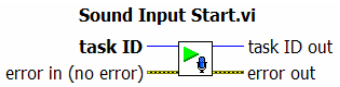
Y=tablou eşantioane în interval [-1, +1] dacă tipul este DBL sau SGL

› **SI Stop**.vi opreşte achiziţia de la Placa Sunet



› **SI Start.vi**: reporneşte achiziţia de la placa de sunet (pune eşantione în bufferul de memorie RAM alocat).

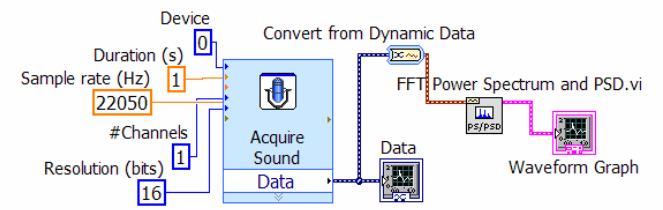
* se apelează numai daca anterior s-a apelat SI Stop.vi



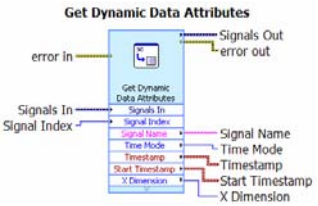
› **SI Clear.vi**: eliberează memoria de sarcina de achiziţie, inclusiv bufferul alocat.

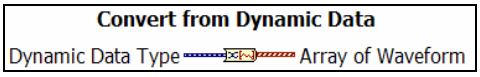
2.2. Aplicație #1. *Achiziție sunet un timp finit* *(1 sau 2 canale)*

Acquire Sound (de la sound device) Paleta Sound/ Funcția predefinită:

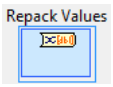


- ⇨ achiziționează semnal de la microfon
- ⇨ **durata** achiziției este finită (exemplu 1s)
- ⇨ Frecvența de eșantionare / preluare eșantioane/ sec.
- ⇨ returnează în **format dinamic** (grafic sau listă de valori)
  Vezi *Get Dynamic Data Atributtes*.
- ⇨ Conversie: *Dynamic Data → tablou de waveforms*





!! Se includ funcții pentru prelucrarea dorită a datelor achiziționate



- ⇨ este șters automat taskul după achiziția semnalului.

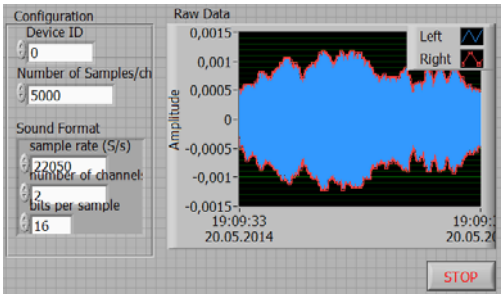
2.3. Aplicație #2. Achiziție timp nedefinit (continuă)

Continuous sound input.vi / 2 canale => Raw Data = 1D array of waveform

LabVIEW 2010\examples\sound2\sound2.llb\Continuous Sound Input.vi

La **o ciclare** achiziționează și afișează 5000 eșantioane.

Ciclul se OPREȘTE dacă: 1.**apare eroare sau**
2.**se apasă STOP**



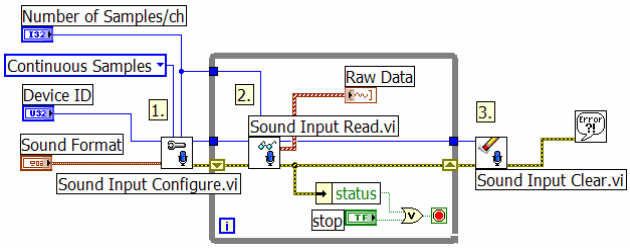
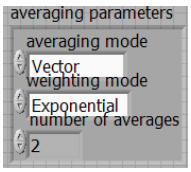
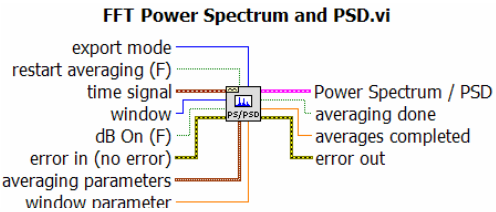


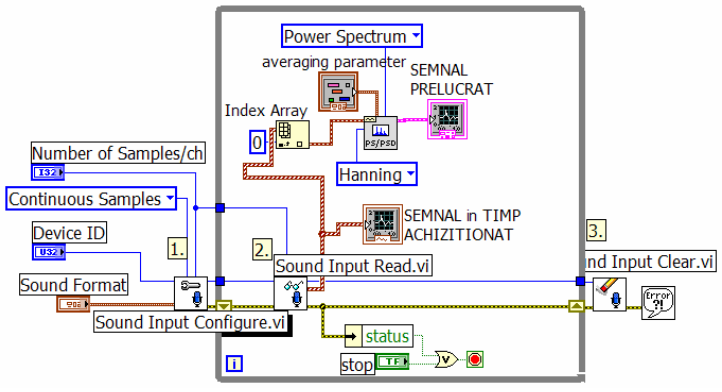
Fig.5 Lview ver.2010. Ex. achiz. continuă PlacaSunet: 1=configurare, 2=citire din buffer, 3=Clear;

Teme:

1. Se adaugă calcul putere spectrală a semnalului,
   - se alege fereastra Hanning,
   - averaging parameters →
2. Se repetă prelucrarea precedentă folosind numai un canal (din tabloul de forme de undă se selectează prima formă de undă...)

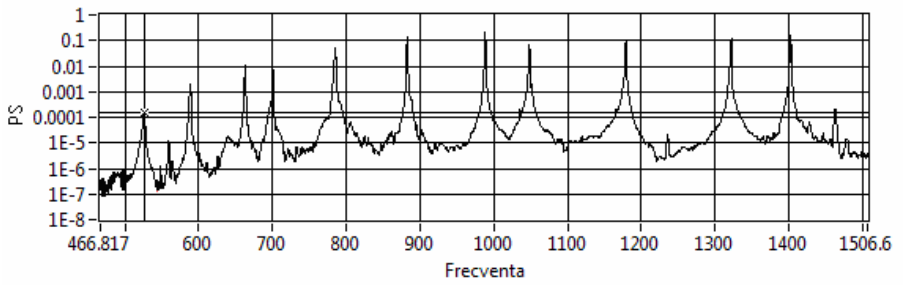




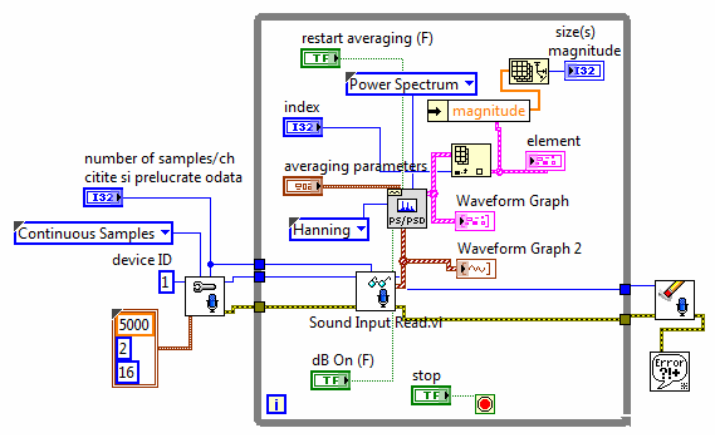


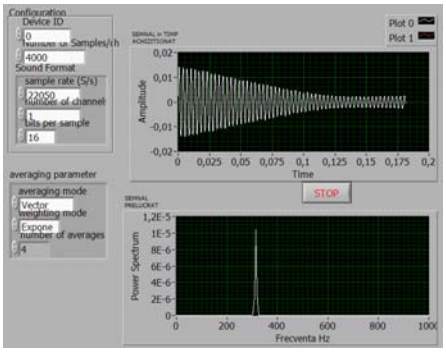
Se măsoară frecvențele unor sunete:
diapazon, pahar sticlă, instrument muzical, bătut din palme etc. (se vor folosi cursori asociați graficelor)

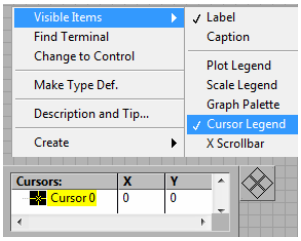
Aplicație: Măsurare frecvențe **Xilofon 12 lamele metalice** (prima lamelă C5)



1.Puterea spectrală măsurată. 2.se identifică abscisa (frecventa) vârfurilor







Cursor mas. frecv. vârfuri

Frecvențe măsurate asociate vârfurilor din puterea spectrală

C5	D	E	F	G	A	B	**C6**	D	E	F	G
527	589	663	701	785	883	988	**1049**	1179	1322	1402	1569

OBS:

1. Fs =2000es/s iar Nr. es / ch =4000 => un ciclu se face la 2 secunde deoarece SI Read.vi așteaptă 2 sec. ca să primească 4000 eș. deoarece sunt preluate numai 2000 eș. pe secundă de la convertor; cele 4000 eșantioane vor intra în PowerSpectrum. Spectrul este actualizat odată la 2 secunde (deci rar) iar df=0.5Hz, iar lățimea spectrului este 0-1000Hz.

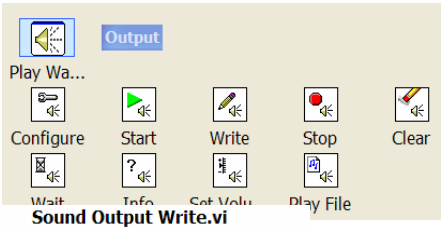
2. Dacă Fs =4000 eș/s iar Nr. eș / ch = 2000 => în 0.5 secunde vin cele 2000eș la SI Read.vi și apoi sunt trimise la PowerSpectrum. Deci vor fi 2 cicluri pe secundă, df=2Hz și spectrul este actualizat de două ori /s; lățimea spectrului este 0-2000Hz

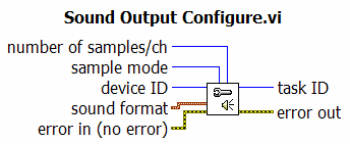
3. Pentru orice Fs și Nr. es / ch frecvența tonului (diapazon) este corect determinată.

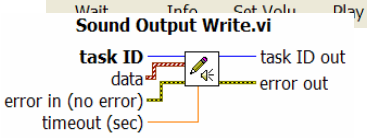
4. df=Fs/N unde Fs este frecvența de eșantionare iar N este numărul de eșantioane analizate deodată.

Frecvența maximă din spectrul rezultat f_{max} =N/2*df = **Fs/2**.

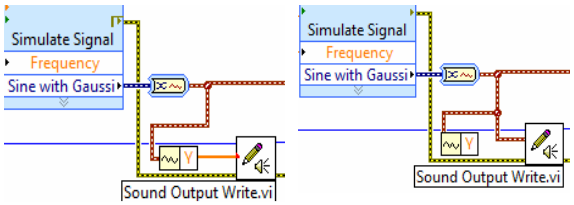
3. Funcţii de bază pentru GENERARE sunet prin Placa de sunet → la difuzoare





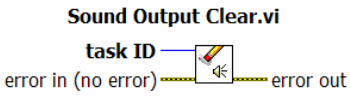


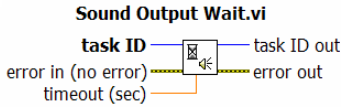
Configurează Placa-sunet pt. generare (convertor digital/analogic) Impune nr. eş./canal alocate în buffer Mod generare= finit sau continuu sound Format= a) rata de trimitere a eşantioanelor 44kHz, 22.kHz sau 11.kHz b) Canale: 1=mono, 2=stereo c) 16 sau 8 biti/ eşantion	Scrie **data** la **dispoz. de ieşire** (în buffer memorie alocat) **data**=tablou de waveforms câte un waveform /canal t0 se neglijează, **dt** se neglijează **Y trimis la SO Write** Y poate fi SGL, DBL cu val [-1...1] U8 [0...255], I16 etc.

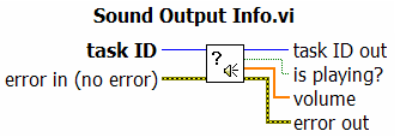


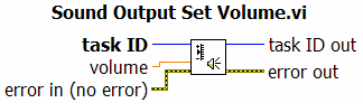
Acelaşi sunet generat deoarece t0 şi dt sunt ignorate ↑

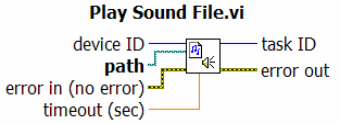
Sound Output Stop.vi task ID, error in (no error) → task ID out, error out * opreşte sunetul = opreşte placa de sunet să ia eşantione din buffer	**Sound Output Start.vi** task ID, error in (no error) → task ID out, error out * Reporneşte generare sunet după oprire cu SO Stop.vi * se foloseste numai după SO Stop.vi



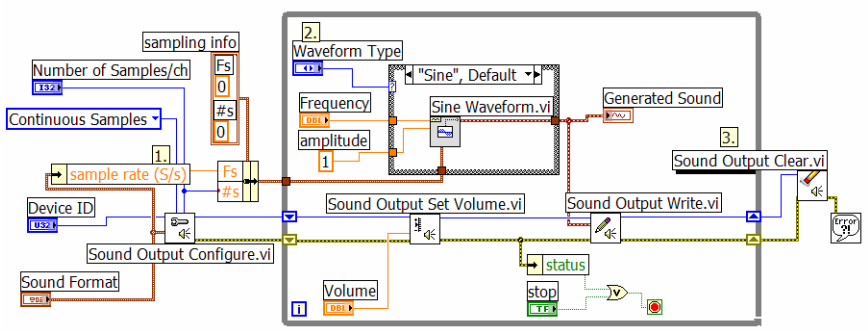




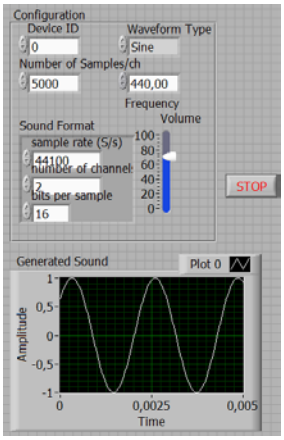




APLICATIE #1 *generare*: LabVIEW 2010\examples\sound2\sound2.llb\
Generate Sound_verMai2015.vi



1.Configurarea Plăcii Sunet pt.:
generare de sunet în mod continuu,
Number of samples/Channel=5000 eş
= (marime buffer RAM)
** = #eş generate de *Sine Waveform.vi*
Sound Format= structură cu 3 câmpuri:
1.sampling rate=**44100 eş/s** (rata de generare =44100 eş. pe sec.)
***setează şi Sine Waveform.vi*
2.canale=**2** (se generează pe ambele canale, stereo),
3.bits per sample=**16** bits (valoarea eşantionului se reprezintă pe 16 biţi).



2.CASE : selectează **tipul semnalului de ieşire**.
Ex: **Sine Waveform.vi**:
- semnalul are frecvenţa=440Hz şi amplitudinea =1
Structura **sampling Info** a funcţiei Sine Waveform :
iniţial (0,0) şi se modifică la (44100 eş/s, 5000 eş) pentru a coincide cu setarea plăcii de sunet. Semnalul sinusoidal ar conţine 440 perioade dacă ar conţine 44100 eşantioane dar fiind numai 5000 eş. va fi alcătuit din proporţional mai puţine perioade de sinus:

44100 eş ... 440 cicluri (Hz)
5000 eş ... x cicluri x=440* 5000 / 44100 = 49.88 cicluri

şi va dura proporţional mai puţin de o secundă:

44100 eş ... 1 sec.
5000 eş ... y sec. y= 5000 / 44100 = 0.113 secunde.

* ciclul while asigură prin generare repetitivă continuu semnal pentru Sound output write.vi

3. Sound Output **Write**= generează sunet de 49.88 cicluri timp de 0.113 secunde prin convertorul digital-analogic spre difuzor

4.Se afişează semnalul în panoul frontal

5. Se repetă paşii 2,3,4 (ciclul While) până la apăsare Stop sau eroare: *STOP if True*
8.82 ciclări pe secundă *(=44100/5000)*
** 44100eş trebuie generate pe sec. dar câte 5000eş la o ciclare*
*8.82 ciclări While *49.88perioade sinus → 440 perioade sinus pe secundă*
6. Se şterge sarcina de scriere la Placa de Sunet apelând SO Clear.vi.

Inlocuiţi controlul frecvenţă cu Horizontal Slide, setaţi limitele la 200Hz şi 10kHz. Ascultaţi în căşti tonuri de frecvenţa dorită în cadrul limitelor stabilite.

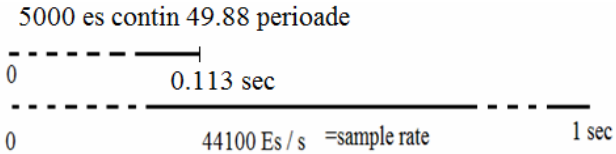


Fig. Sine waveform.vi: 49.88 cicluri compuse din 5000eş la frecvenţa 440 Hz

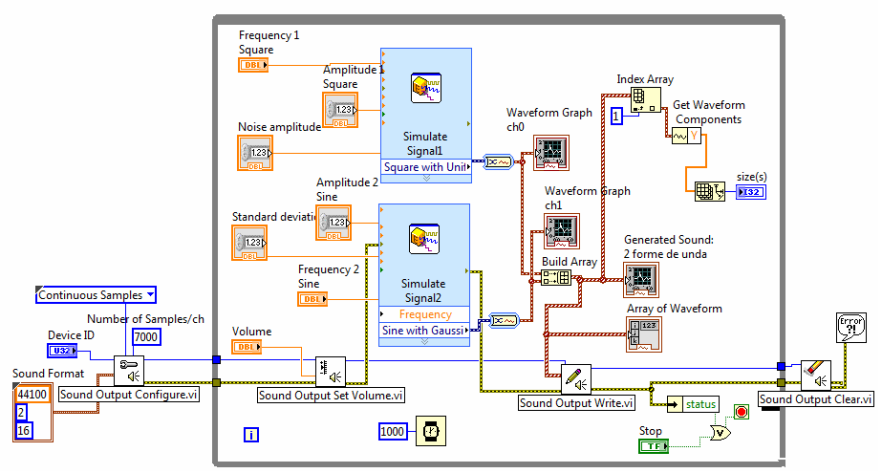
APLICAŢIE #2** generare: **Tablou de 2 waveforms (2 canale) -> Sound Output Write.vi
* sunt generate cu *Simulate Signal.vi* două forme de undă:
Sine + noise → canal 0

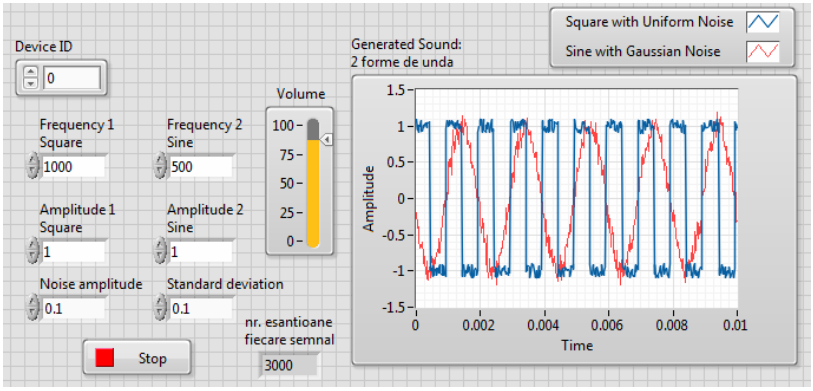
Square + noise → canal 1

* fiecare waveform conține **3000 eşantioane** scrise la **Sample rate: 44100** Hz =>

=> durată waveform= 0.068sec, $(\frac{1}{44100} \cdot 3000)$

* SoundOutputWrite.vi primeşte tabloul de 2 forme de undă (generat cu build array) pentru 2 canale





* afişarea în PF este cu zoom pe durata 0...0.01 sec. pentru a distinge semnalele,

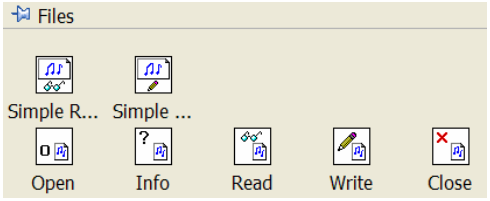
* dacă se reduce Sampling Rate de la 44100 la 22050 sunetul generat va dura mai mult şi se va auzi un ton mai jos deoarece acelaşi semnal (şi număr de eşantioane) este trimis la Placa de sunet cu o rată mai mică. Obs:frecvenţa generată se schimbă deoarece nu este corelare între SO Config.vi/ Sound Format şi generarea semnalelor (acestea nu folosesc Sampling Info (Fs, #s))

* In fereastra *'Generated Sound 2 forme de unda*' se afişează aceleaşi semnale fiindcă semnalele generate de funcţiile Simulate signal.vi sunt aceleaşi.

*Semnalul dreptunghiular se aude în difuzorul de pe primul canal iar semn. sinusoidal la difuzorul conectat canalului al doilea.

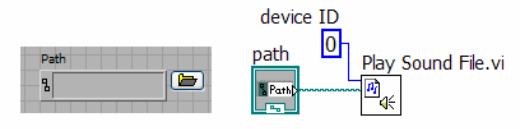
*volumul la difuzoare se reglează prin funcţia SO Set Volume.vi (în limitele 0...100) şi prin amplitudinea funcţiilor periodice în intervalul 0 şi 1. Dacă amplitudinea semnalului sinusoidal creşte peste 1 volumul nu creşte.

4. Fişiere



4.1. Exemplu: Funcţia **Play Sound File.vi**

trimite un fişier .wav (**chimes.wav**) salvat pe disc →la placa de sunet (boxe) Lime green.



4.2. Conținutul fișierului .wav poate fi :

1. trimis la difuzor cu ***Play Sound File**.vi* şi
2. citit/vizualizat grafic array de waveform (conținutul) cu ***Sound File Read Simple.vi***

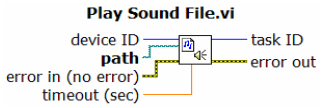
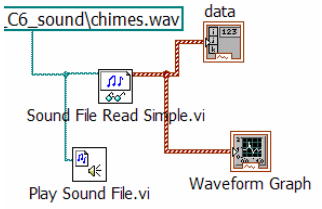
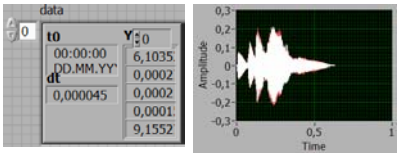
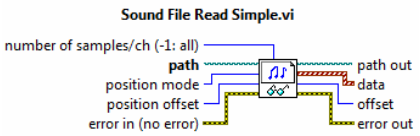


Fig: Play .wave file







Obs: dt=0,000045sec

x

22100eş/sec = 1

dt x rata[Hz] =1

*** Sound File Read Simple.vi** → citeşte din fişier wav, **(Nu de la microfon)**

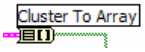
Citeşte întregul fişier .wav specificat prin calea la fişier sau numai primele n eşantioane.

*returnează:

data= *datele într-un tablou* de forme de undă

5. Aplicație: Simulate a Telephone.vi :

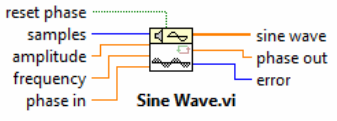
› calculeaza o pereche de tonuri (inalt, jos) pentru tasta apăsată şi generează un sunet corespunzător pentru validare şi recunoaşterea sonoră a tastei.

- aplicația ramâne în buclă while până la apăsarea butonului de "**stop".**
- La fiecare iterație registrul de transfer este inițializat cu STRUCTURA de constante logice 4x3 False,
- Dacă se apasă o tastă în PFrontal numai acel câmp al structurii va avea valoarea "True"
- ***Cluster To Array*** 

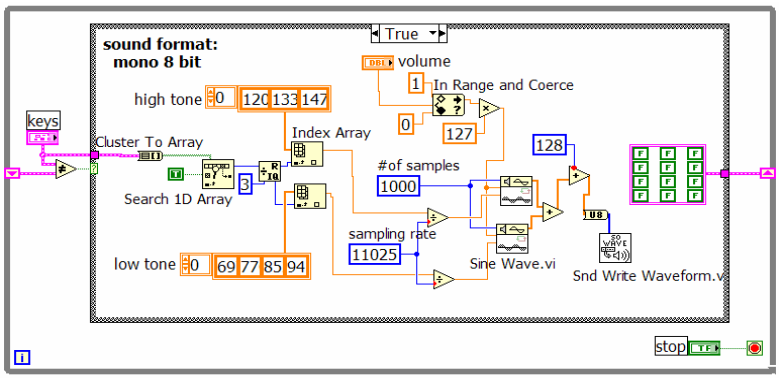
transformă structura "keys" (câmpuri logice) într-un şir=TABLOU **1D** de tip logic,
- ***Search 1D Array*** returnează numărul de ordine a tastei apăsate (T=true) în şirul 1D logic de taste (toate valorile sunt F mai puțin una care este T)
  exemplu tastezi 7 => index 6
- Impărțirea cu rest determină coloana (R=0,1,2) şi linia (IQ=0,1,2,3) tastei apăsate;
  Ex: tasta 7 (poziția indice 6) =>
  6:3 => R (coloana)=**0**, IQ (linia) =**2**
- Un **Index Array.vi** extrage valoarea frecvenței din:
  constanta *Tablou **frecvențe înalte*** (high tone)
- Alt Index Array.vi extrage altă valoare de frecvență din:
  constanta *Tablou **frecvențe joase*** (low tone)
- Se generează folosind două funcții ***Sine Wave.vi***:

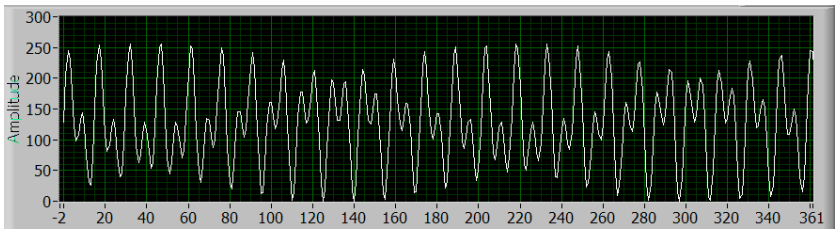
*două sinusoide (tonuri de frecvențe diferite)

*amplitudinea sinusoidelor este între 0 şi 127 fiind dată de controlul/slide “volume”
*fiecare sinusoidă gener. 1000 eşantioane
*sinusoidele se însumează + offset de 128 şi conversie la U8 => sunet pt. SO Wave



- **Snd Write Waveform.vi** trimite/generează la difuzor sunetul specific tastei.

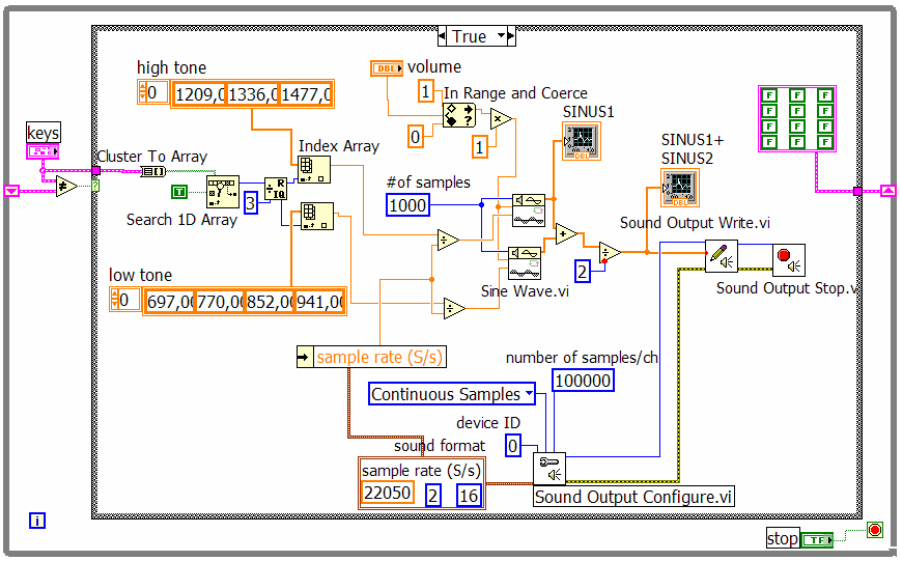




Semnalul trimis la *SO Wave.vi* la apăsarea tastei 2

# of samples = 1000, numărul de eşantioane din semnalul Sine Wave (implicit 128),
amplitude = implicit =1.0, Aici comandată de intrarea “volume”,
f este frecvența Sinusului în unități normalizate cycles/sample
(implicit 1cycle/128 samples sau **0.0078125 cycles/sample**).
Dacă frecventa =1/10 => 10 samples/ciclu => 12.8 perioade daca samples=128.

Varianta #2:



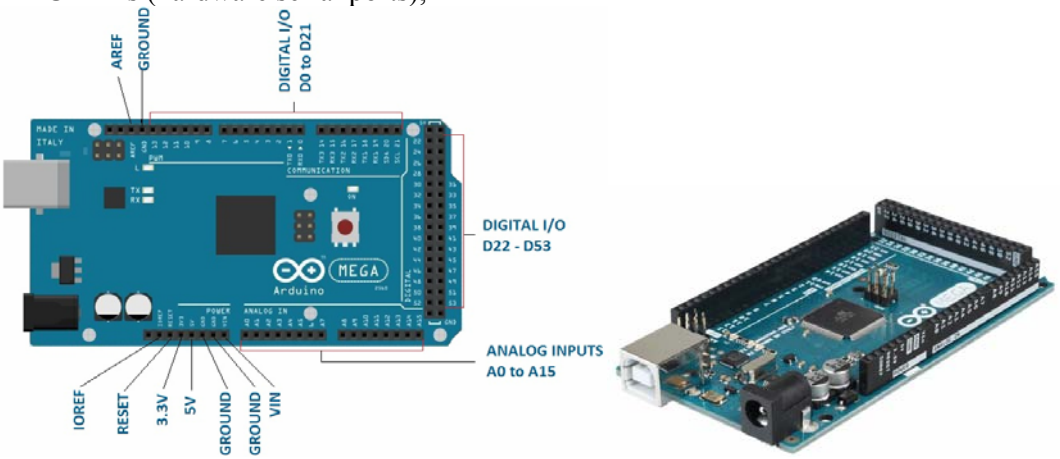
Varianta #2: gen. sunetului cu Sound Output Configure.vi, SO Write.vi, SO Stop.vi
** SO Config.vi se poate apela înafata ciclului While,
** se adaugă SO Clear.vi la ieşire din ciclul While.

II. Utilizare Placa Arduino MEGA 2560 din Labview

1. Descriere Arduino MEGA 2560

placă cu microcontroler construit pe baza procesorului **ATmega2560**.
* 8-bit RISC single-chip microcontrollers.

- are **54 pini digital input/output**,
  din care **15 pot fi folosiți ca ieșiri PWM** (pulse width modulation):
  pinii: **2 - 13** (pinii 0 și 1 sunt pentru tx și rx)
  +pinii **44, 45, 46** sunt deasemenea pini output 8-bit PWM comandați
  (cu funcția analogWrite())
- **16 analog inputs A0,..., A15**,
- 4 UARTs (hardware serial ports),



- un cristal oscillator (clock speed) la 16 MHz,
- un conector USB → permite conectare la calculator prin cablu USB
- a power jack → permite alimentare de la un adaptor AC to DC sau de la baterie
- un buton de reset.

LED Built-in : pinul 13 (L)

- Mega 2560 board este compatibilă cu Uno și altele.
- 5 pini Ground sau masă (GND)

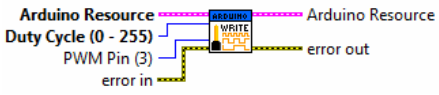
2. Arduino Intrări/Ieșiri semnale digitale și analogice

Plăcile Arduino prezintă intrări și ieșiri pentru semnal digital și numai intrări pentru semnal analogic. Semnalul **analogic** poate lua orice valoare într-un interval. Semnalul **digital** poate avea numai două valori: HIGH și LOW.

Arduino are convertor analog →digital ADC (built-in) pentru a măsura semnale analogice deci pentru a converti semnalul analogic în valori digitale. Pentru aceasta se folosește funcția Arduino IDE: **analogRead**(pin) sau funcția Labview: **Analog Read Pin.vi**. Aceste funcții convertesc valoarea de tensiune generată de senzor disponibilă la pinul analogic (între valorile 0 și 5V sau 0 și 3.3 volți în funcție de placa Arduino) în valoare digitală întreagă între 0 și 1023. Deci convertorul este pe 10biți (2^10=1024). Rezoluția de citire a semnalului analogic este 5 V/ 1024 units => 0.0049 volts (4.9 mV) per unit.

Pentru plăcile cu procesor ATmega (UNO, Nano, Mini, Mega), o citire din semnal analogic se face în 100 microsecunde (0.0001 s) astfel rata de citire este de maxim 10,000 Eșantioane/secundă.

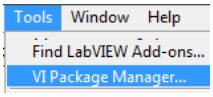
Plăcile arduino nu au convertor Digital→Analog DAC dar pot parțial suplini neajunsul prin semnal PWM (pulse-width modulate) care este un semnal digital dreptunghiular de o anumită frecvență (490Hz, 500Hz, 980 etc., în funcție de placă). Astfel este folosită funcția **analogWrite**(pin, value) unde: **pin** =numărul pinului folosit pentru ieșirea semnalului PWM iar **value** este un număr proporțional cu **duty cycle** a semnalului dreptunghiular (pentru value = 0 se generează semnal LOW tot timpul ; pentru value = 255, semnalul generat este HIGH tot timpul. Labview folosește din paleta Arduino funcția PWM Write Pin.vi în mod similar.



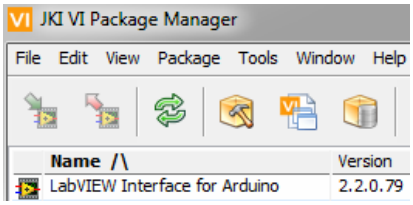
Arduino Uno: pinii PWM sunt: 2,..,7.
Arduino Mega pinii PWM sunt: 2,...,13 și 44,45, 46.

3. Interfața LABVIEW cu ARDUINO toolkit

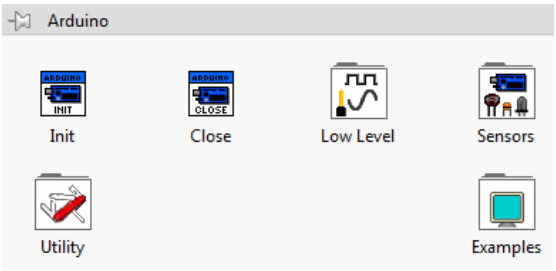
1. Se lansează
   Tools/***VI Package Manager***



2. Se instalează ***Labview Interface for Arduino*** toolkit LIFA din Tools/***VI Package Manager*** prin download de pe www.ni.com



3. Se observă/instalează toolkit Arduino

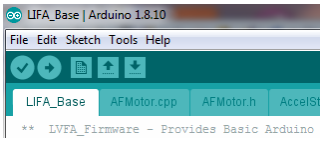


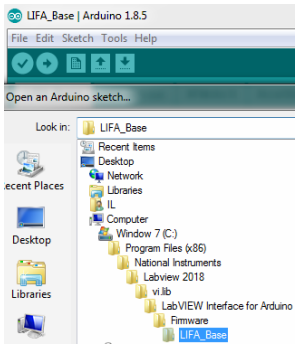
Aplicația #1. Comandă LED încorporat de pe placa Arduino
LEDul este ataşat la pinul digital 13

1. Se lansează Arduino IDE
   (instalarea se face prin download de pe www.arduico.cc)

2. Se conectează Arduino Mega 2560 prin USB la laptop.

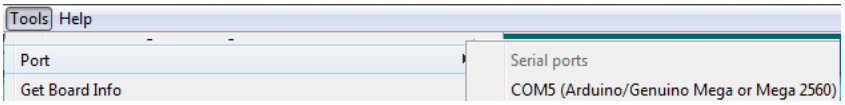
3. Din mediul Arduino se încarcă fişierul LIFA_Base. *C:\Program Files\National Instruments\LabVIEW Instruments\...\vi.lib\LabVIEW Interface for Arduino\Firmware***LIFA_Base**



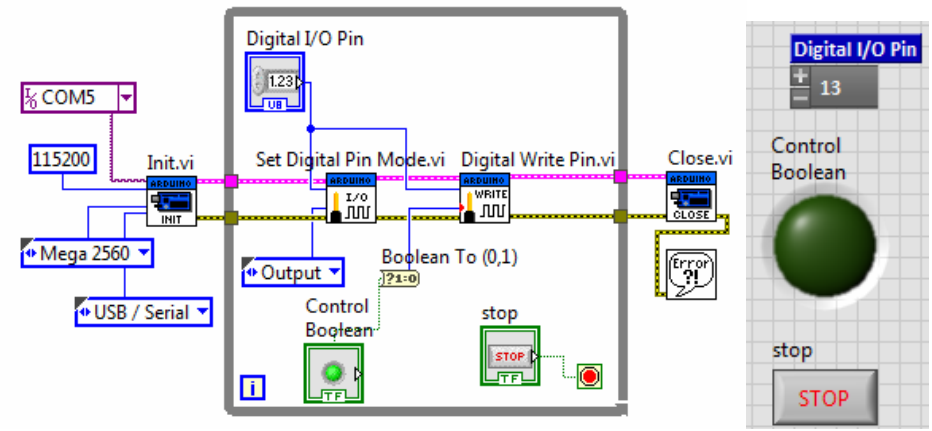


4.Alege placa: Tools\Board \Arduino/ **Genuino** Mega or Mega 2560.
Alege Portul: de exemplu ... COM5
(COM1, COM2 sunt deja alocate pentru hard serial...)

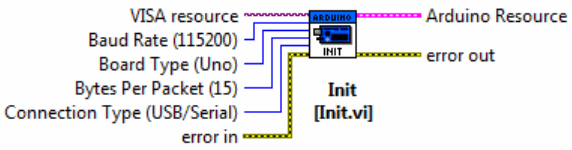
5.Click **Upload** buton pentru încărcare firmware în Arduino.
- Se observă ledurile RX şi TX de pe placă blinking după care
- Arduino IDE răspunde cu **Done Uploading** dacă încărcarea firmware este cu succes în mediul Arduino.
- Din acest moment se poate folosi Arduino Mega 2560 cu LabVIEW Interface for Arduino.

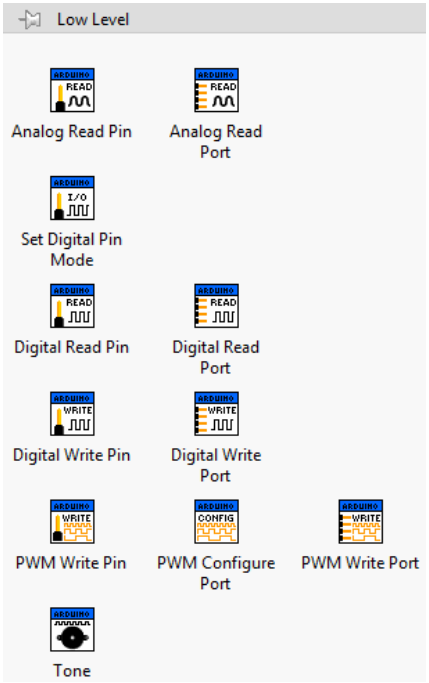


6. Se construieşte aplicația Labview PF+Diagrama folosind paleta Arduino din Labview.
7. Se rulează aplicația Labview; se aprinde/stinge LED pe placă prin Control Boolean de pe PF.

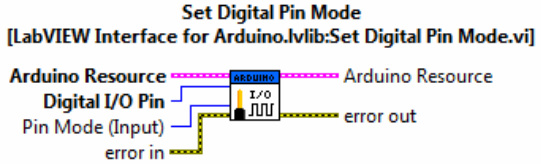


Init.vi → inițializează legătura cu placa Arduino aflată sub interfața Labview interface with Arduino.

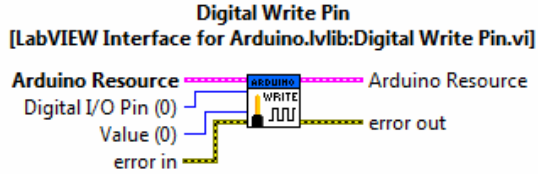




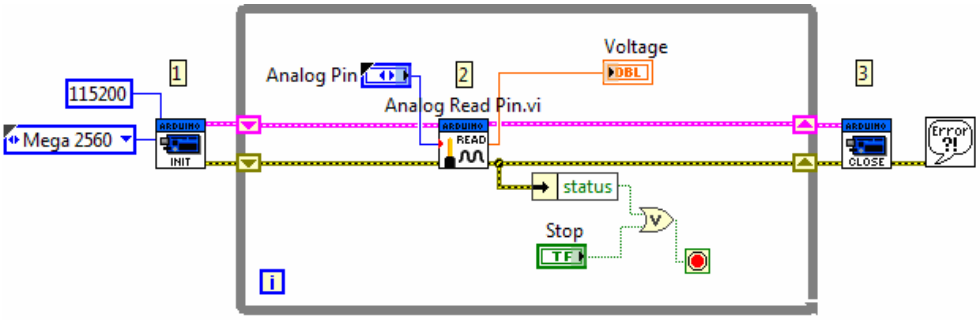
Set Digital Pin Mode.vi setează pinul specificat din plaja D0-D13 pentru input sau output.



Digital Write Pin.vi → scrie valoarea specificată prin intrarea Value, la pinul indicat din plaja D0-D13. In prealabil pinul trebuie setat pe *output* cu funcția Set Digital Pin Mode.



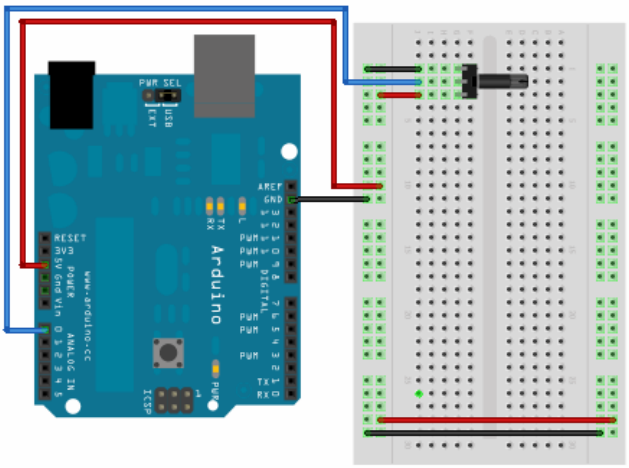
2. Citire semnal analogic - potențiometru (EU_Analog Read Pin Example.vi)



Potențiometrul se conectează astfel:

1. Picior mijloc la un pin din plaja 0..5 din grupul de pini **ANALOG IN** (fir albastru),
2. Pin lateral potențiometru la masă **GRD** fir negru,
3. Pin de la celălalt capăt/lateral potențiometru se conectează la **pin 5V** (Arduino).

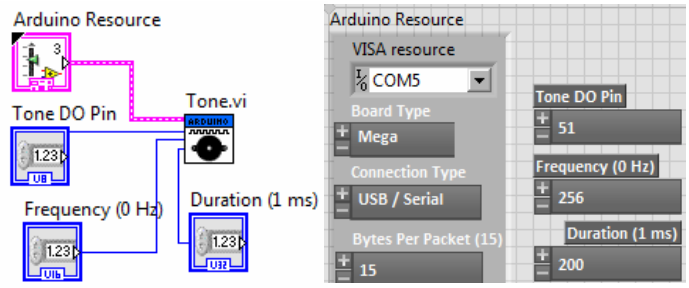
Rotind ştift potențiometru se modifică tensiunea citită de pe potențiometru.

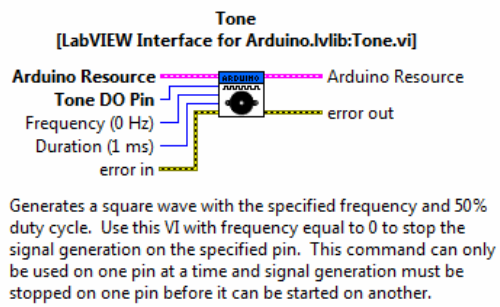


3. Generare un ton cu funcția **Tone.vi** din Arduino/ Low level/ Tone.vi

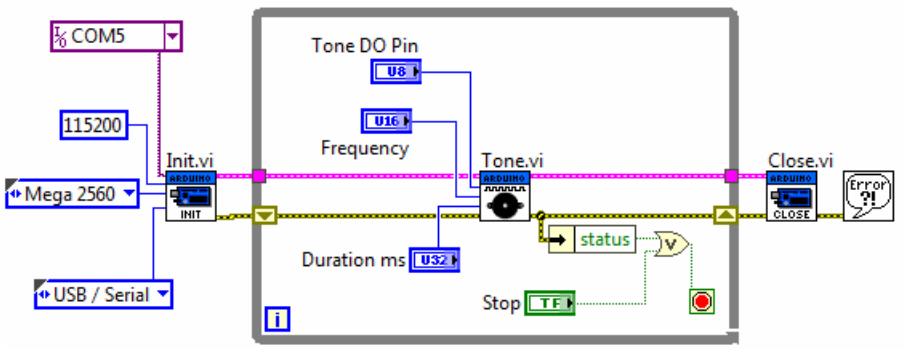
Generează un TON

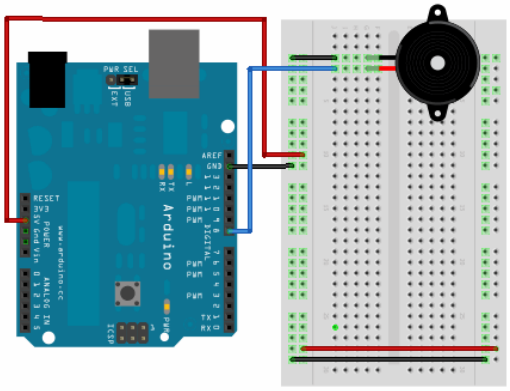
1. Se conectează placa Mega prin USB la Laptop.
2. Se selectează Board Type: Mega; VISA resource: COM5; USB/Serial
3. Se conectează Buzzer cu fir roşu la pin digital 8 sau 32 sau 22 sau **45** (Pin PWM sau nu) şi cu fir negru la GND (lângă Aref).
4. Aplicația generează un ton la frecvența impusă în PF de durată 0.2sec.



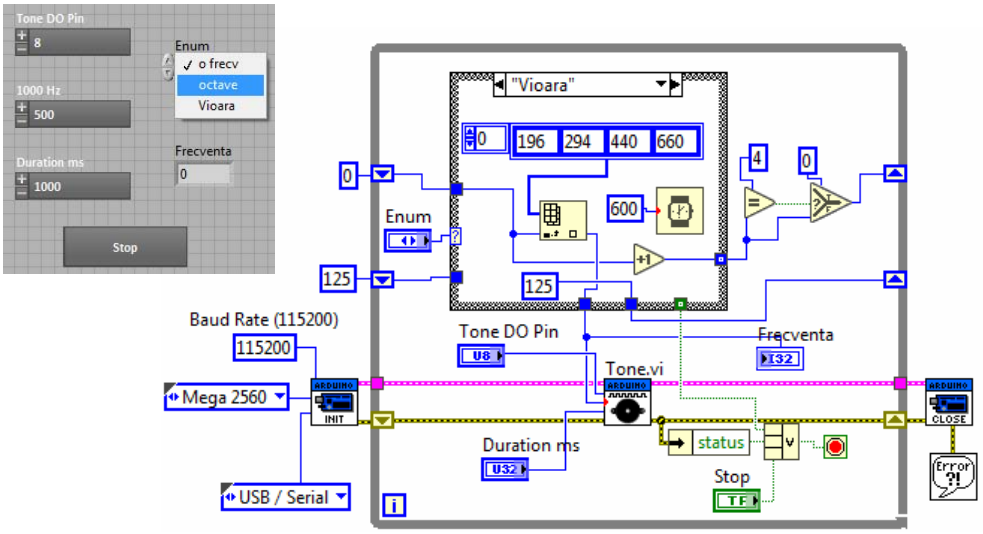


3.2.Aplicația generează un ton la frecvența impusă în PF de durată 1sec, ciclic. Este folosit KEPO KPR-G3010-6250 Piezo Transducer 1.3kHz 35mm din figură conectat la GND digital cu fir negru şi la pin digital 40 (sau altul) cu fir roşu.





3.3. Varianta **cu frecvențe** comandate _EU_Arduino_Tone.vi



Completați aplicația cu selector Enum.
Selectați 'Vioara': se aud repetitiv cele 4 note asociate corzilor viorii.
Selectați 'O frecventă': se aude 1000Hz
Selectați 'Octave': se aud sunete care își dublează frecvența pornind de la 125Hz până la 8000Hz inclusiv.

4. Citire 2 sau mai mulți senzori:

1.Senzor lumina brick SEN-LUM-03

- sesizează nivelul de iluminare al mediului.

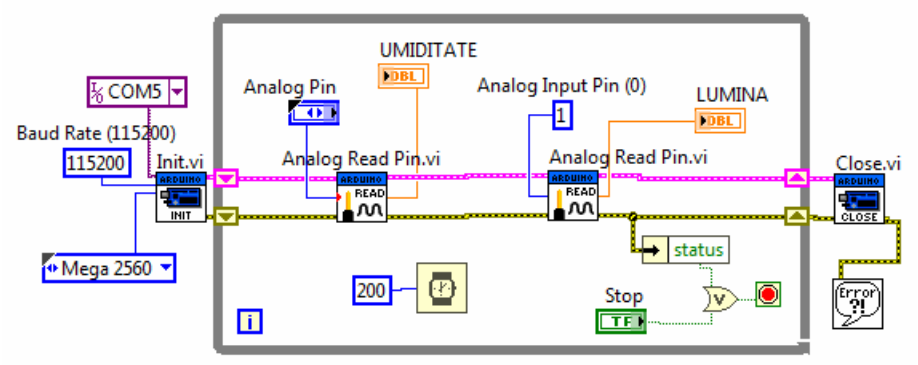
Valoarea iluminării variază liniar intre 0 si 1023. Senzorul nu este calibrat deci nu se citeşte o mărime corespunzătoare unei unități de măsură.
Pinul de semnal (OUT) se conectează la un **pin analogic** al placii Arduino;
Pinul de alimentare (VCC) se conectează la pinul 5V al placii Arduino;
Pinul de masa (GND) → la pinul GND al placii Arduino

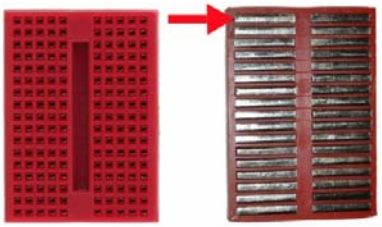
2.Senzor umiditate brick SEN-VRM-08

- sesizează nivelul de umiditate al mediului.

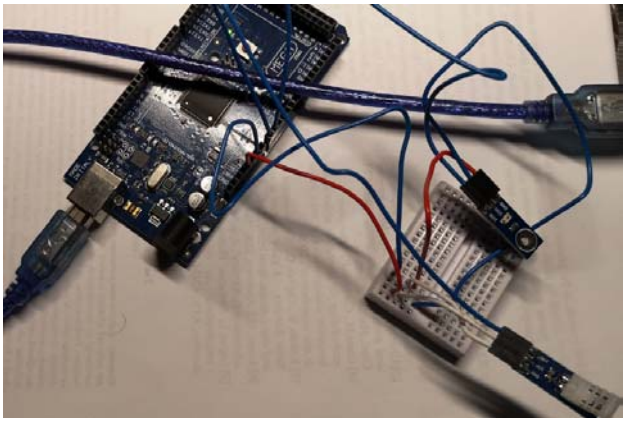
Intr-o camera obisnuită, valoarea citita pe portul analogic variaza intre 900 (mediu saturat cu vapori de apa) si 300 (foarte uscat).

- senzorul nu este calibrat
Pinul de semnal OUT (stanga) se conecteaza la un **pin analogic** al Arduino





Fecare senzor prezintă 3 pini: VCC, GND şi OUT. Se foloseşte un **mini breadboard** (mBB) pentru conectarea senzorilor. Cinci (5) intrări (orificii) aşezate pe o linie sunt conectate între ele electric. Sunt astfel 17 linii pe o parte şi 17 linii de cealaltă parte a canalului de răcire a mBB.



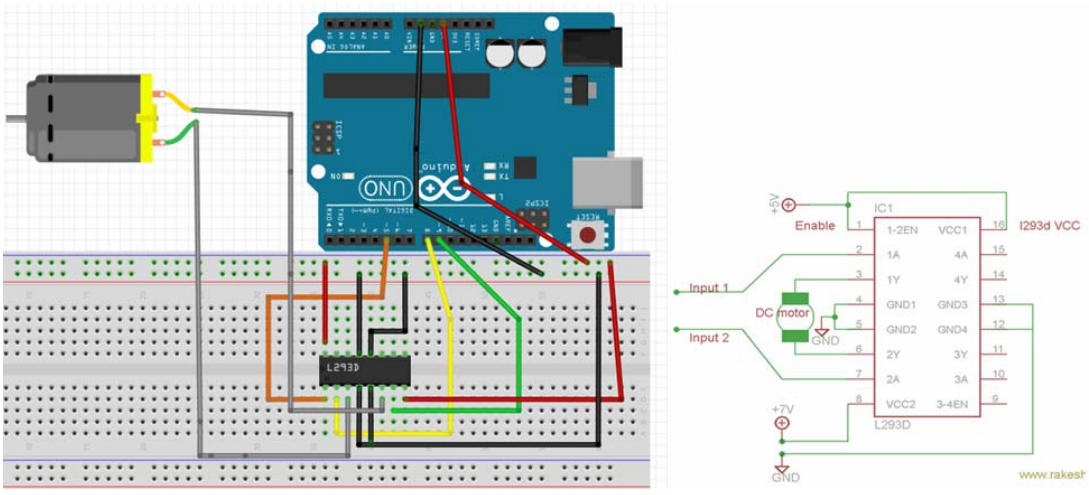
O linie se conectează la sursa de 5 volţi V5 a plăcii (zona POWER). La această linie se conectează pinii VCC (central) a celor doi senzori.

O altă linie se conectează la un GND placă (zona Power). De pe aceeaşi linie se conectează cei doi senzori cu pinii GND (marginal).

Pinii de semnal OUT a senzorilor se conectează la Portul analogic indice 0 şi indice 1 a plăcii Arduino zona ANALOG IN.

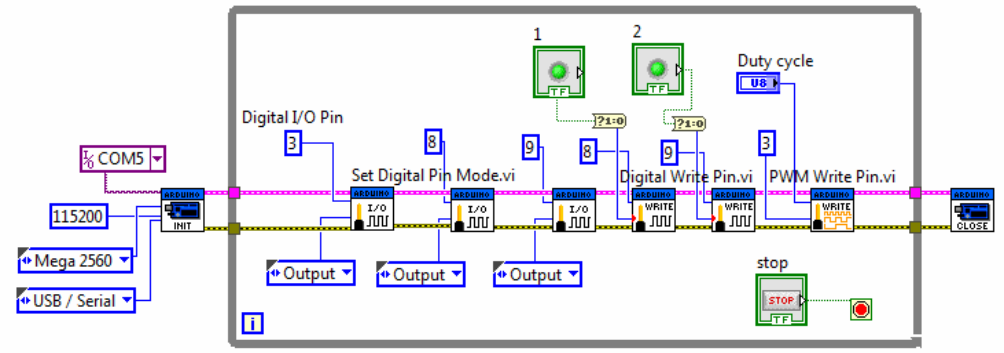
Se citesc în aceeaşi buclă pinii OUT a doi senzori cel de Umiditate şi unul de Lumină.

5. Control Sens şi Turaţie MOTOR DC cu IC motor driver tip L293D

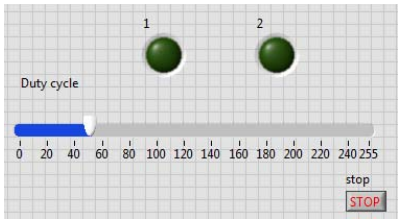


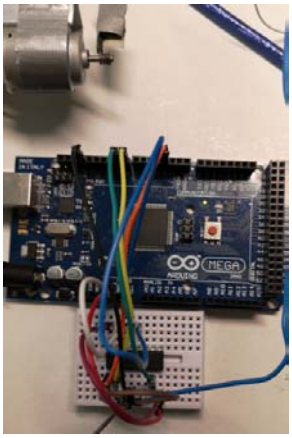
Arduino pinul 5 conectat la L293D pinul ENA1 prin PWM comanda viteza variabila tensiune variabila din control DIAL
Arduino pinii 8,9 conectati la L293D pinii 2 , 7 =input1 si input2 pentru control sens rotatie

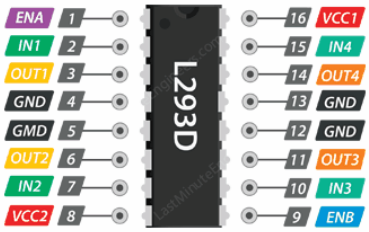
Pinii 12, 13 (latura fără motor) se pot deconecta de la GND-ul placii, fără a perturba funcționarea motorului.



Sens1 rotație motor:
led1=On, led2=Off
Sens2 rotație motor:
led1=Off, led2=On
Motor oprit: ambele leduri pe **Off**
Reglaj turație: pin digital PWM, variație Duty cycle 0 ←→ 255







Conectare Arduino MEGA şi L293D: pinii 8, 9 la IN1(2) respectiv IN2(7); pinul 45 la ENA1 (reglaj turație). Pinul GND Arduino (Power) la pinii 4, 5 (GND L293D). Pinul 5V Arduino (Power) la VCC1 (16) (pentru alimentare L293D) şi de asemenea la VCC2 (8) L293D pentru alimentare motor.
Conectare L293D şi Motor: Pinii OUT1 (3), OUT2(6) la bornele motorului de curent continuu.