**ROMÂNIA**

**MINISTERUL APĂRĂRII NAŢIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**Specializarea: Calculatoare și sisteme informatice pentru apărare și securitate națională**

  
  
***Proiect microprocesoare – Senzor de sunet A***

**Studenți:**

**Sd. Sg. Maj. Anda MARTINIUC**

**Sd. Plt. Adj. Dragoș IOANA**

**București**

**2022**

Cuprins

[1. Scopul proiectului 2](#_Toc92936170)

[2. Modul de conectare 2](#_Toc92936171)

[3. Configurare module 4](#_Toc92936172)

[4. Rularea proiectului 10](#_Toc92936173)

[5. Probleme întâmpinate 12](#_Toc92936174)

[6. Referințe 12](#_Toc92936175)

# Scopul proiectului

Prezenta lucrare practică are rolul de a exemplifica utilizarea unui senzor analogic, anume un senzor analogic de sunet DFRobot v2.2, prin programarea modulelor și setarea regiștrilor implicați, având ca rezultat final afișarea unui grafic de detecție în timp real.

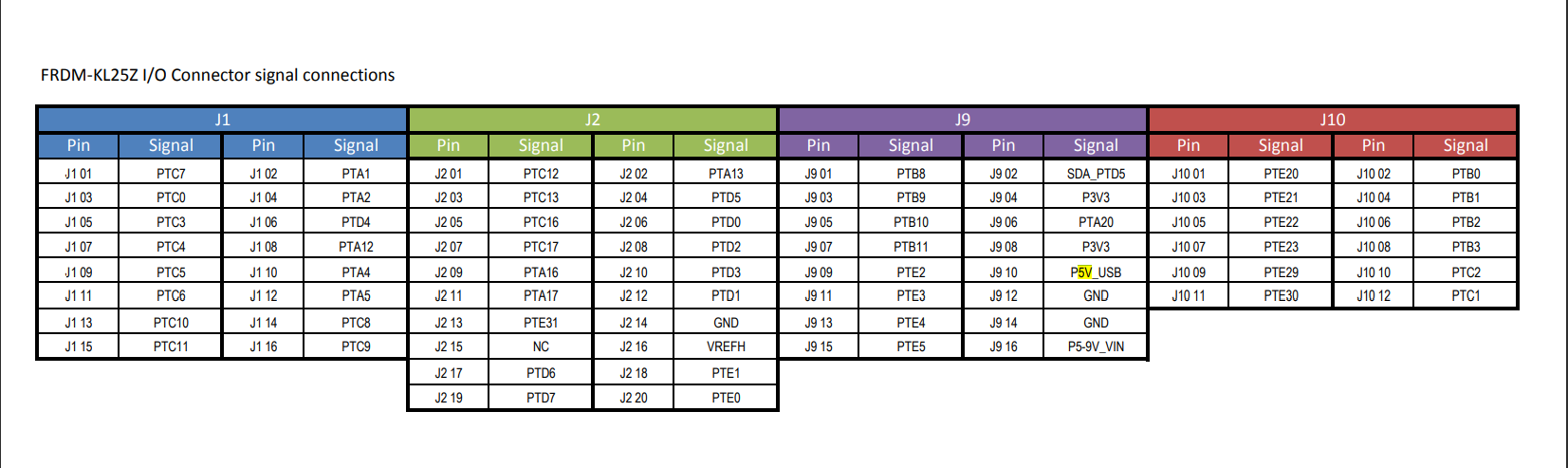
În ceea ce privește aplicabilitatea acestuia în contexte de uz cotidian, pot fi încadrate următoarele exemple:

* Crearea de sisteme electronice de consum (precum telefoanele, calculatoarele, sistemele muzicale, etc.)
* Sisteme de securitate și monitorizare, precum alarme antiefracție, alarme pentru uși, etc.
* Automatizarea locuinței, spre ex. iluminarea casei prin detectarea de fluierat sau bătăi de palme în loc de rotirea sau interacțiunea manuală cu întrerupătorul luminii
* Recunoașterea sunetului ambiental și recunoașterea nivelului de sunet, pentru adunarea de date prelucrate mai apoi pentru a putea fi folosite de o rețea neuronala sau etc.

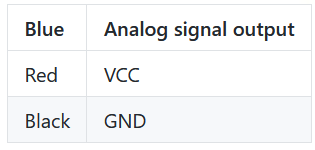
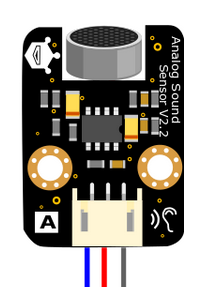
# Modul de conectare

Conform diagramelor din *Figura 1*, respectiv din *Figura 3* ce detaliază pin-urile plăcuței KL25Z, am folosit:

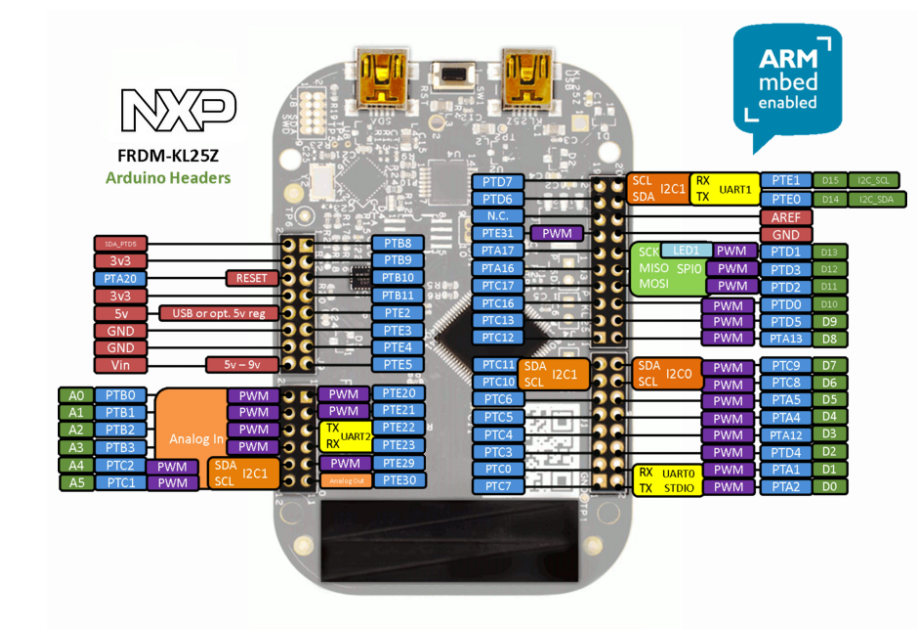
* **J9 8(3.3V)**  pentru alimentarea senzorului analogic
* **J9 14(GND)** pentru ground-ul senzorului analogic
* **J10 2(PTB0)** pentru semnalul de ieșire analogic



Figură Tabel de corespondență al pinilor

****

Figură Diagrama de conexiuni



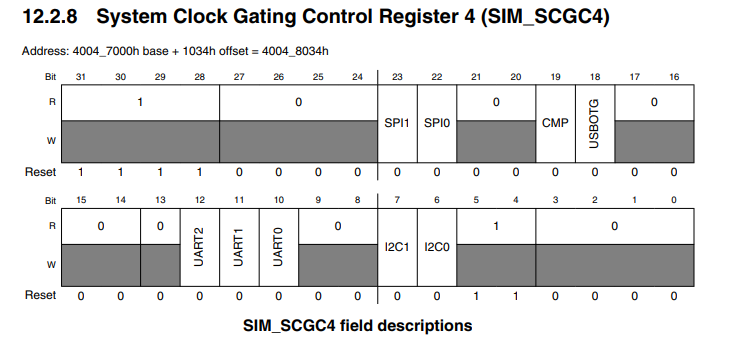
Figură Pini și module asociate

# Configurare module

* UART (Reference Manual)

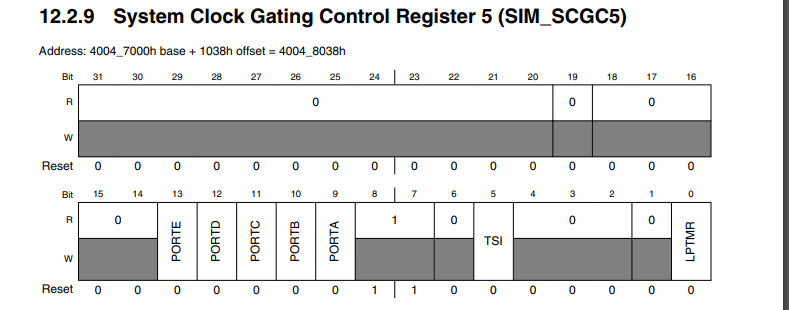
Modulul UART a fost configurat pentru realizarea comunicației prin intermediul portului serial între codul programului și script-ul de Matlab. Fluxul aplicației este început de întreruperea software a modulului ADC preia date de la senzor ( 1 octet), pe care îl transmite prin UART cu un baud rate setat de 115200 și ulterior datele sunt citite din Matlab într-un vector ce urmează a fi afișat și resetat.

Regiștrii configurați in cadrul modulului UART sunt:

* SIM\_SCGC4 – pentru a activa ceasul modulului UART0 (am setat bitul folosind masca SIM\_SCGC4\_UART0\_MASK furnizată de Keil - bitul 10 corespunde canalului UART0)

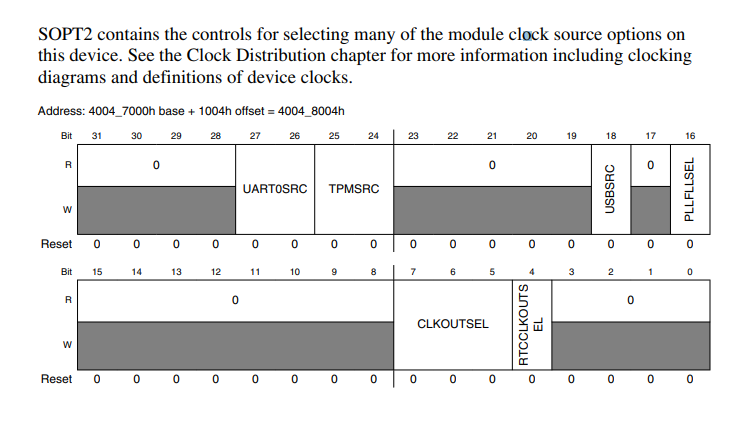
Figură SIM\_SCGC4

* SIM\_SCGC5 – pentru a activa ceasul modulului PORTA (am setat bitul folosind masca SIM\_SCGC5\_PORTA\_MASK furnizată de Keil - bitul 9 corespunde canalului PORTA)



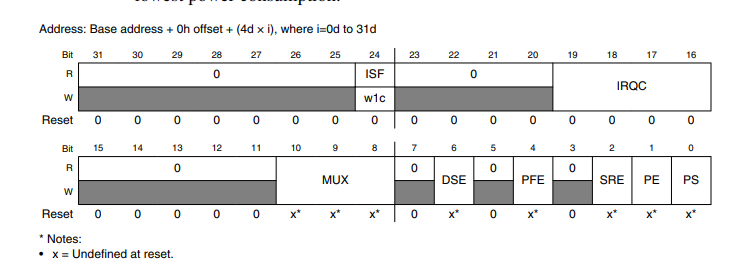
Figură SIM\_SCGC5

* SIM\_SOPT2 – folosim acest registru pentru a seta sursa de ceas pentru UART0, pentru care am setat biții 26-27 cu valoarea 01

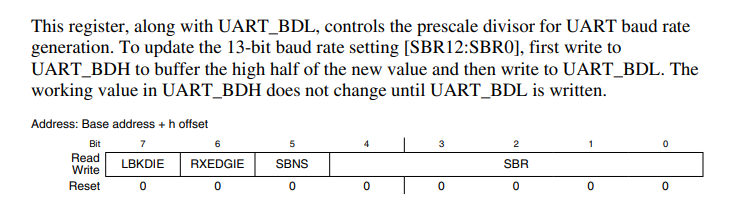


Figură SIM\_SOPT2

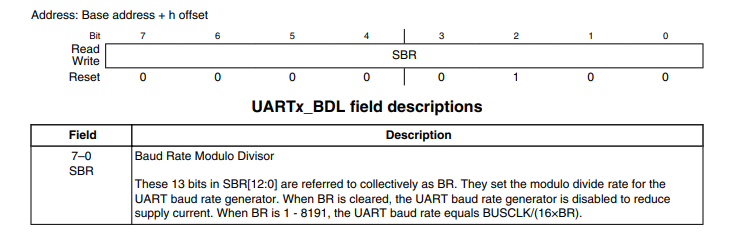
* PORTA\_PCR1 si PORTA\_PCR2 –setam biții de multiplexare la 010 (biții 10 si 8) deoarece acești pini sunt utilizați pentru UART0(PTA1 la citire și PTA2 la transmisie)



Figură PORTA\_PCR

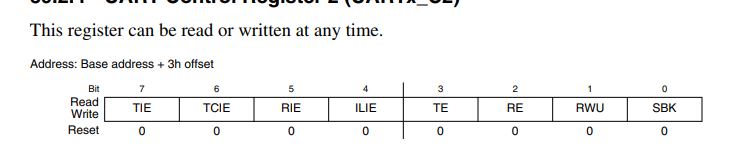
* UART0\_BDL si UART0\_BDH - pentru setarea baud rate-ului de 115200. 

Figură UART0\_BDH



Figură UART0\_BDL

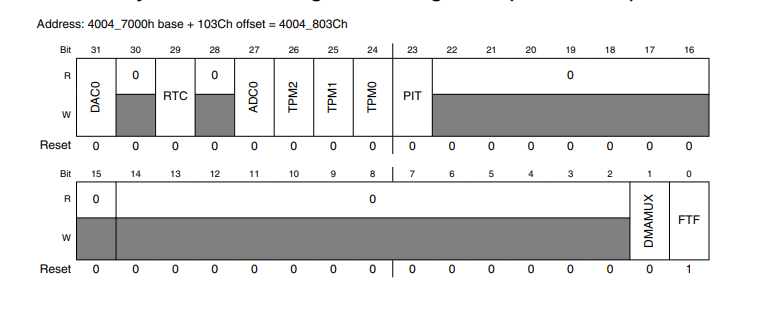
* UART0\_C2 – pentru a controla întreruperile pe UART0. Activăm atât întreruperile de citire, cât și de transmisie.

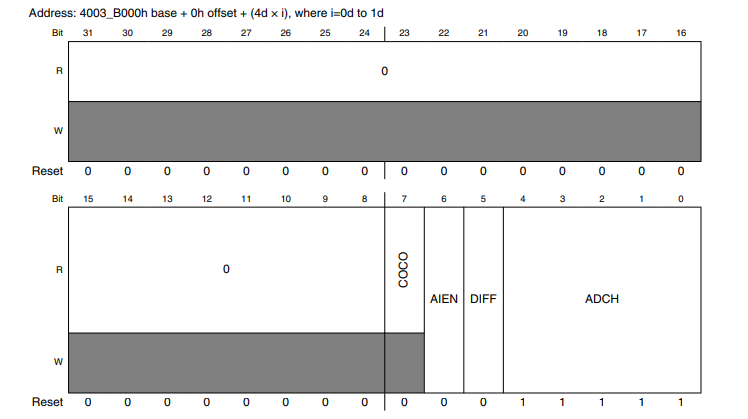


Figură UART0\_C2

* ADC ( Analogic Digital Converter)

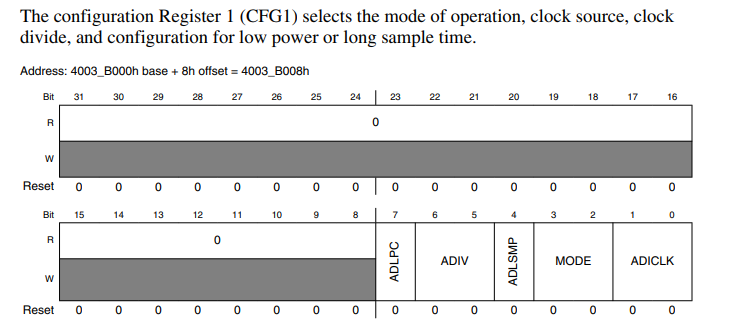
Pentru acest proiect am folosit pinul PTB0 care este corespunzător pentru ADC0\_SE8. Pe acest pin se face conversie pe canalul 8.Regiștrii folosiți în cadrul modulului ADC sunt:

* SIM\_SCGC6 –pentru a activa ADC0 (setăm bitul 27)
* ADC0\_SC1A – pentru status și pentru activarea întreruperilor ( bitul 6 Interrupt Enable )



Figură ADC0\_SC1A

* ADC0\_CFG1 pentru a seta rezoluția convertorului prin punerea biților de MODE pe 00 (DIFF este 0, deci conversia este single-ended pe 8 biți) și pentru a seta rata de împărțire la 8 prin punerea biților din ADIV pe 11

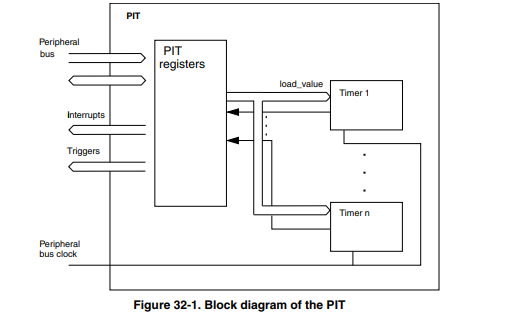


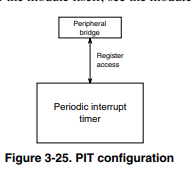
Figură ADC0\_CFG1

* ADC0\_RA – se pune rezultatul conversiei.

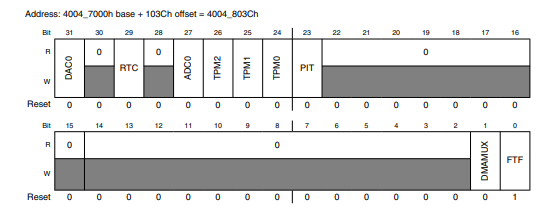
Datele transmise aici(Rn) conțin rezultatul unei conversii ADC a canalului selectat de starea și registrul de control al canalului corespunzător (SC1A:SC1n). Pentru fiecare registru de control al stării și canalului, există un registru de rezultat al datelor corespunzător

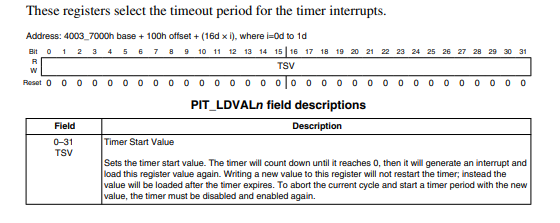
PIT ( Periodic Interrupt Timer )



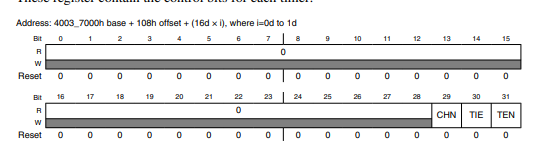


Motivația folosirii modulului de PIT rezultă din timpul foarte scurt la care senzorul transmite date pe portul ADC( aprox. 9 us).Eu îmi definesc o întrerupere proprie de PIT la intervale de 10ms și în cadrul aplicației o integrez prin folosirea unei variabile pe 32 de biți în care voi aduna fiecare citire rezultată din întreruperea hardware. După acel interval, voi folosi o regulă de trei simplă pentru a converti acea sumă într-o valoare pe8 biți pe care o putem transmite pe UART.

* SIM\_SCGC6 – pentru a activa PIT (pentru a activa ceasul modulului PIT (am setat bitul folosind masca SIM\_SCGC6\_PIT\_MASK furnizată de Keil - bitul 23 corespunde canalului PIT) 
* PIT\_MCR- activează și dezactivează timer-ul PIT (Bitul 30 – Module Disable: 0- Enable /1-Disable)
* PIT\_LDVAL1 – setează valoarea de start a timer-ului-> Fig.0.17 evidențiază conținutul registrului PIT\_LDVAL1

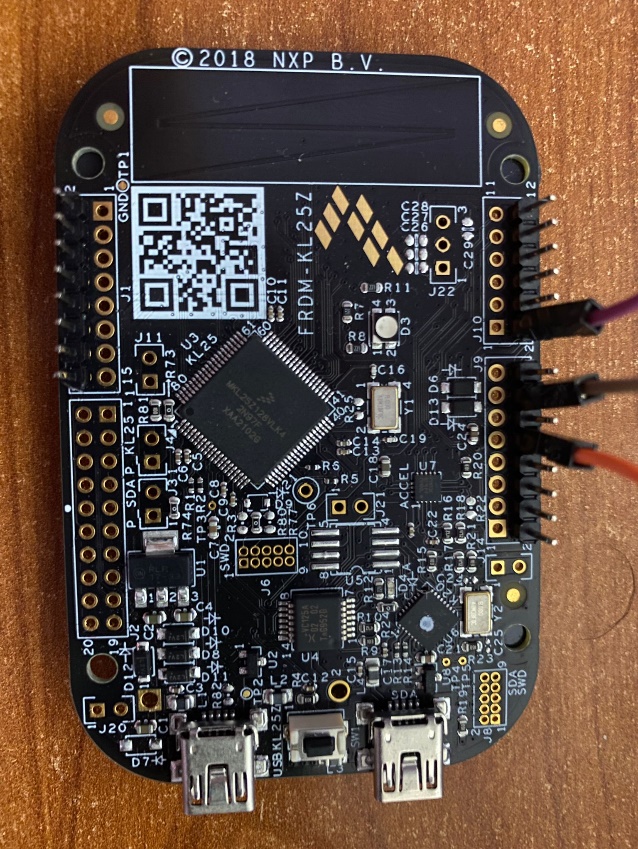


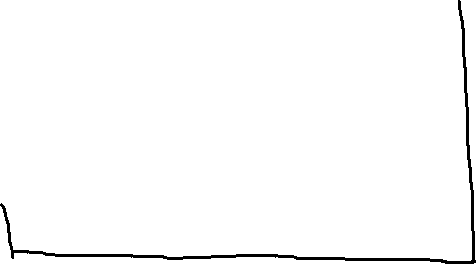
Figură PIT\_LDVAL1

* PIT\_TCTRL1-setam time interrupt enable (bitul 30) 

Figură PIT\_TCTRL1

# Rularea proiectului

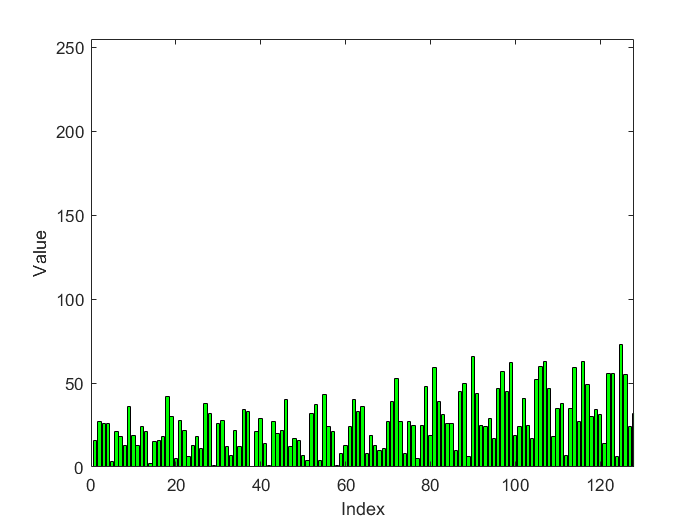
****



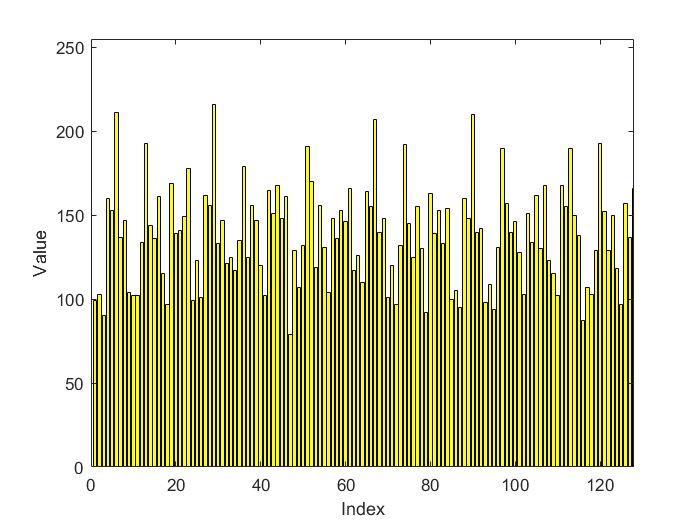
Figură 15 Schema practică a lucrării

După conectarea senzorului la plăcuță, conectăm plăcuța la laptop prin intermediul cablului USB, care se introduce în plăcuță pe portul SDA. Pentru a încărca programul pe plăcuță vom face *Rebuild* și *Load* in Keil. Fișierele sursă ce vor fi compilate și rulate au fost încărcate pe Github la adresa <https://github.com/undacmic/FRDM-KL25Z-Real-time-sound-detection> .

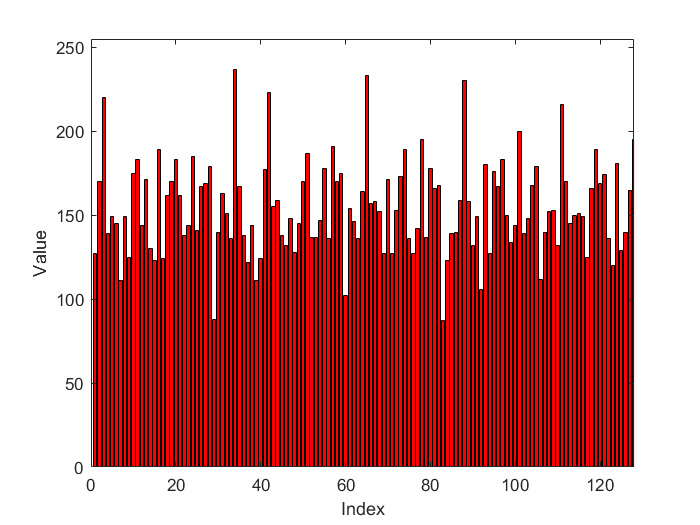
Vom porni ulterior script-ul de Matlab care preia valorile transmise pe comunicația serială, le adaugă într-un vector pe care îl afișează diferit, în funcție de intensitatea sunetului înregistrată de senzor, codificată pe culori astfel



Figură Valori între 0-50



Figură Valori între 50-175



Figură Valori peste 175

# Probleme întâmpinate

* Afișarea unui grafic în timp real format din bare de grosime configurabilă, paralele cu axa oY și de lungime egală cu intensitatea.

**Rezolvare: Identificarea funcției bar și parametrizarea acesteia (setând culoarea barelor graficului în funcție de intervalul de valori în care se situează citirea curentă)**

# Referințe

* <https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-kl14-kl15-kl24-kl25-mcus:FRDM-KL25Z>
* <http://www.pemicro.com/opensda/pe_tools.cfm>
* <http://www.pemicro.com/opensda/index.cfm>
* <https://os.mbed.com/platforms/KL25Z/>
* <https://www.openhacks.com/uploadsproductos/frdm-kl25z_pinouts__rev_1.0_.pdf>
* <https://www.keil.com/appnotes/files/apnt_232.pdf>