Kauno technologijos universitetas
Elektros ir elektronikos fakultetas

MEMS mikrofonas

4 Laboratorinis darbas

Žygimantas Marma, EMEI-2 gr.
Studentas

Doc. Šarūnas Kilius
Dėstytojas

Kaunas, 2023

Turinys

Kaunas, 2023

Darbo tikslas: Susipažinti su MEMS mikrofonais ir atlikti pateiktos programos analizę.

Laboratorinio darbo uždaviniai:

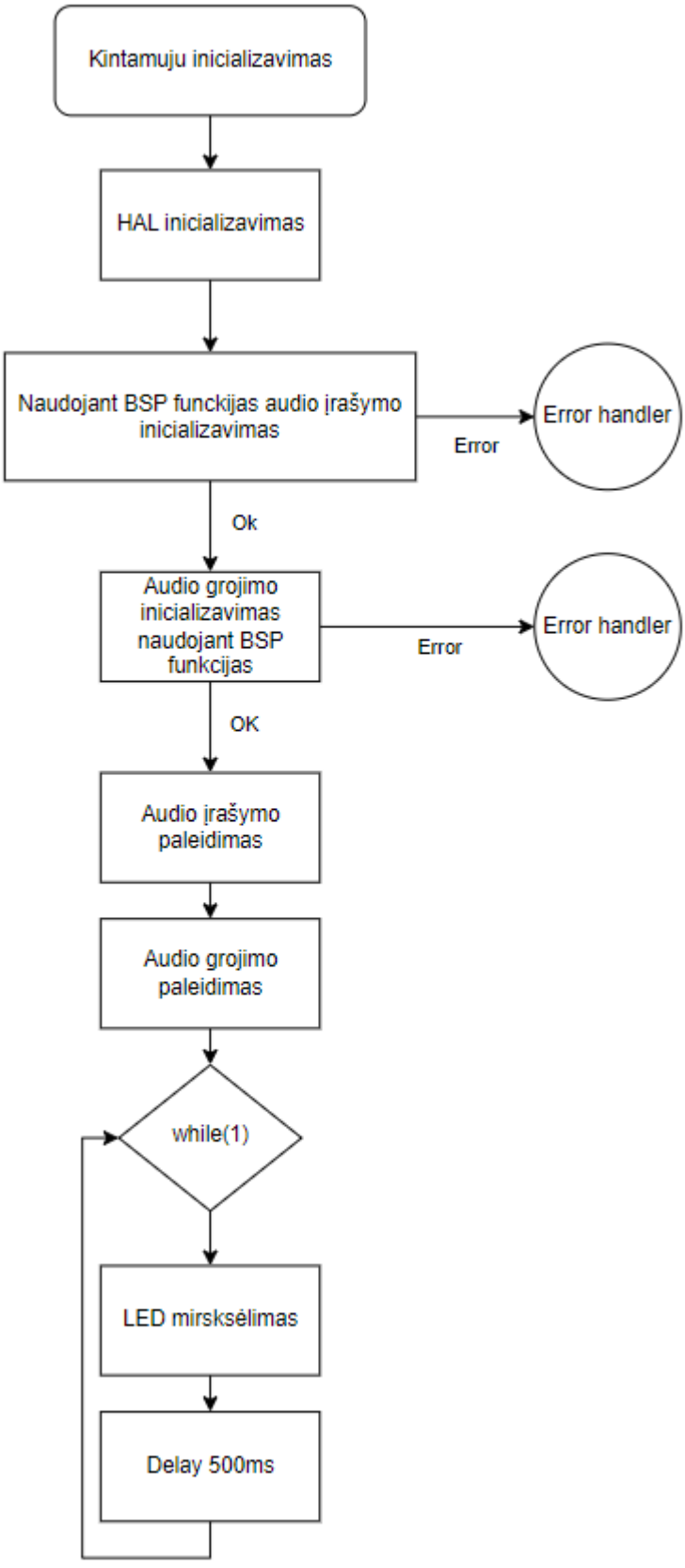
1. Išanalizuoti pateiktą programą ir sukurti programos veikimo algoritmą.
2. Perduodamam garsui pritaikykite vieną iš žinomų filtrų arba garso efektų.

1. Programos analizė

1.1. Programos veikimo algoritmas

Pateikto laboratorinio darbo programa susideda palaipsniui iš šio algoritmo (taip pat atvaizduoto 1 pav.):

1) Kintamųjų inicializavimas:
   - Inicializuojami visi reikalingi kintamieji, kurie bus naudojami algoritme.
2) HAL inicijavimas (Hardware Abstraction Layer):
   - Inicializuojamas abstrakcijos sluoksnis, kuris suteikia abstrakciją aparatūros prieigai, padedant izoliuoti programinį kodą nuo konkretaus aparatūros.
3) BSP funkcijų naudojimas audio įrašymui inicializuoti:
   - Iškviečiamos BSP funkcijos, kurios susijusios su audio įrašymo įrenginio inicializavimu.
4) Audio grojimo inicializavimas:
   - Inicializuojamas audio grojimo mechanizmas, paruošiant sistemą atkurti garsą.
5) Audio įrašymo inicializavimas:
   - Inicializuojamas audio įrašymo mechanizmas, paruošiant sistemą įrašyti garsą.
6) While(1) ciklas (begalinis ciklas):
   - Nepaliaujamai vykdomas begalinis ciklas.
7) LED mirksėjimas:
   - Įvykdomas veiksmas, skirtas mirksėti LED.
8) 500 ms laiko trukmės laukimas (Delay 500ms):
   - Programa laukia 500 milisekundžių, prieš pereinant į kitą iteraciją.

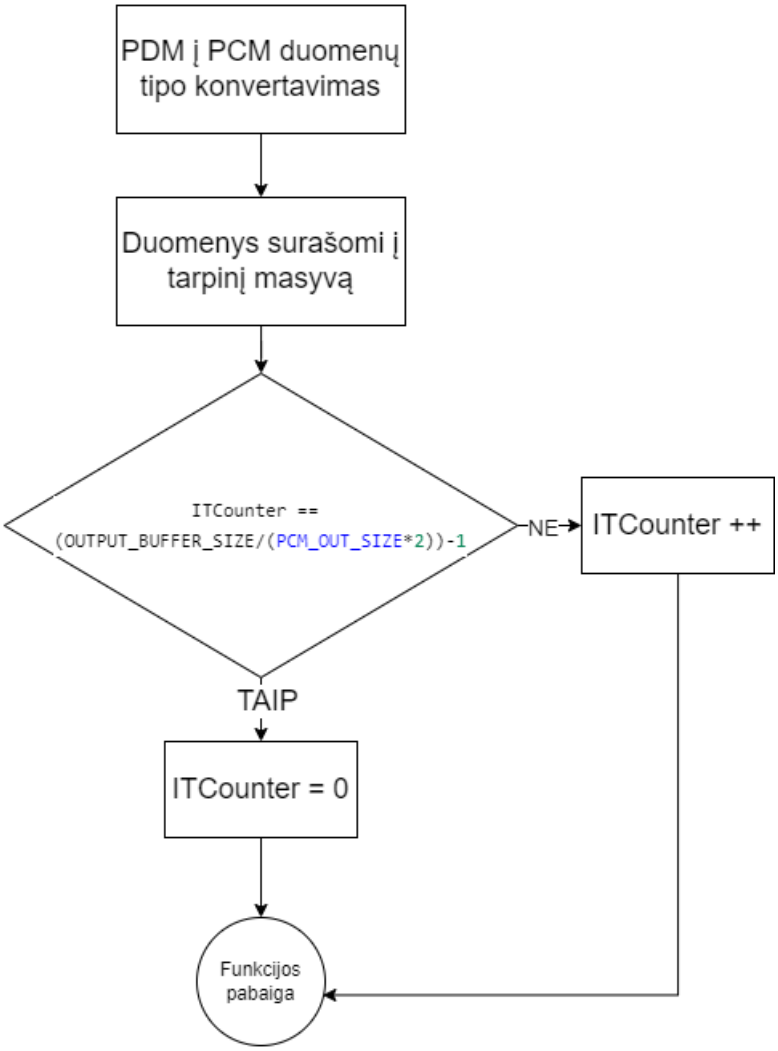


1 pav. Pagrindinės programos veikimo algoritmas

Tačiau esminės sistemos funkcijos yra atliekamos atgalinio iškvietimo funkcijose – callback.

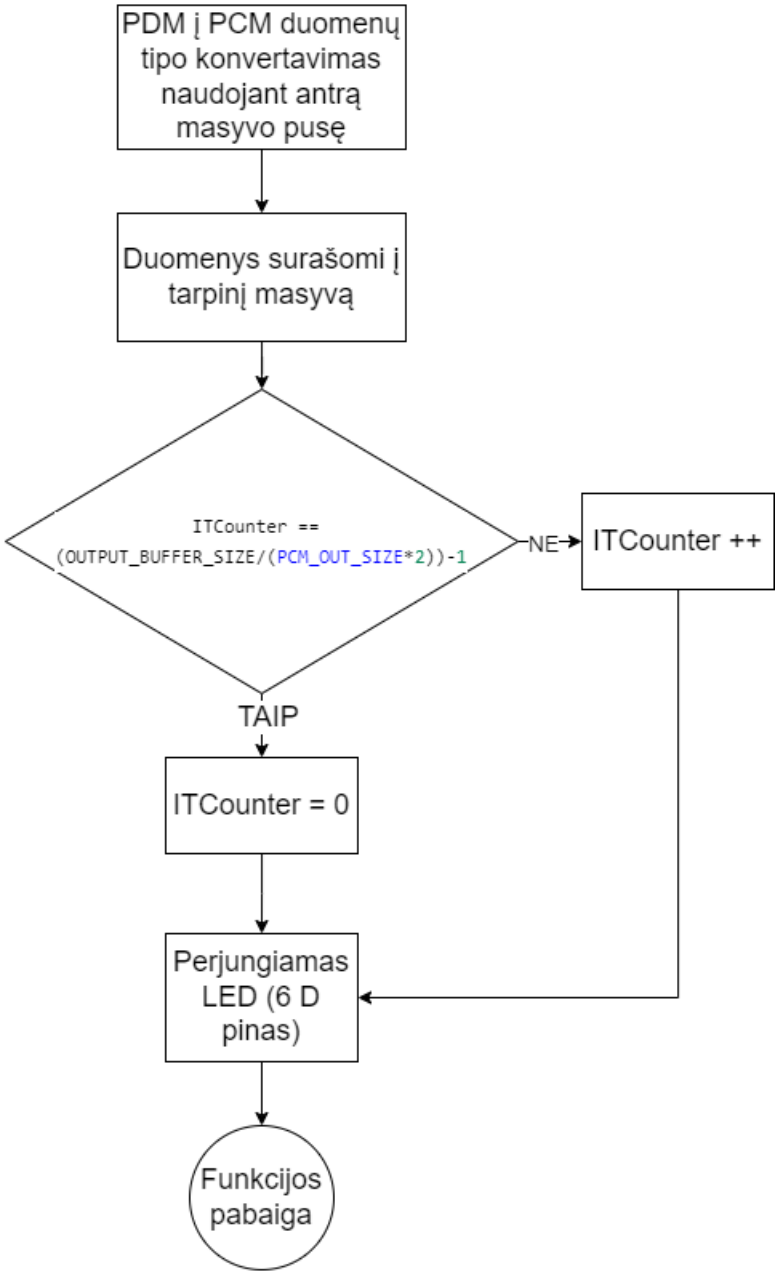
Laboratorinio darbo programinėje įrangoje yra du atitinkami garso įvesties (audio input) apdorojimo metodai, kurie yra iškviečiami kaip pertraukimų (callbacks) funkcijos. Šie metodai yra iškviečiami įvykus tam tikriems įvykiams – kai buferis yra perpus užpildytas (HalfTransfer) arba kai perduodamas pilnas buferis (TransferComplete).

BSP_AUDIO_IN_HalfTransfer_CallBack(void) funkcija yra iškviečiama, kai buferio pusė yra užpildyta. Pirmiausia vyksta PDM (Pulse Density Modulation) į PCM (Pulse Code Modulation) duomenų konvertavimas naudojant funkciją BSP_AUDIO_IN_PDMToPCM. Tada PCM duomenys kopijuojami į vidinį buferį OutputBuffer. ITCounter kintamasis yra naudojamas nustatyti, kur dalis buferio buvo užpildyta. Jei buvo užpildyta viskas, ITCounter yra nustatomas į nulį, kitaip didinamas vienetu. Tai vizualiai parodoma 2 pav. BSP_AUDIO_IN_HalfTransfer_CallBack funkcijos algoritmas



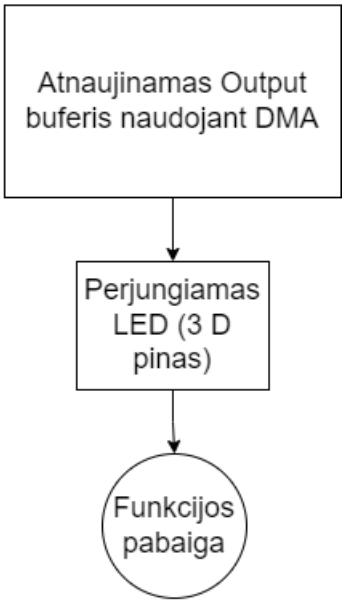
2 pav. BSP_AUDIO_IN_HalfTransfer_CallBack funkcijos algoritmas

BSP_AUDIO_IN_TransferComplete_CallBack(void) funkcija yra iškviečiama, kai visas buferis yra užpildomas. Kaip ir ankstesnėje funkcijoje, vyksta PDM į PCM duomenų konvertavimas ir kopijavimas į vidinį buferį. Po to patikrinama, ar buferis pilnas, ir nustatomas ITCounter kintamasis kaip ir pirmoje funkcijoje. Galiausiai ši funkcija įjungia arba išjungia LED šviesos diodą (HAL_GPIO_TogglePin), signalizuodama apie apdorojimo būsena. Tai vizualiai parodoma 3 pav. BSP_AUDIO_IN_TransferComplete_CallBack funkcijos algoritmas



3 pav. BSP_AUDIO_IN_TransferComplete_CallBack funkcijos algoritmas

Garso grojimo callback funkcija `BSP_AUDIO_OUT_TransferComplete_CallBack` yra iškviečiama, kai garso išvedimo buferis yra visiškai užpildytas ir reikalingas naujas buferis. BSP_AUDIO_OUT_ChangeBuffer funkcija keičia išvedimo buferio rodyklę į pradinę poziciją, leidžiant naujiems garso duomenims užpildyti buferį. HAL_GPIO_TogglePin funkcija įjungia arba išjungia LED šviesos diodą. Taigi, ši funkcija užtikrina, kad garso išvedimo buferis būtų nuolat atnaujinamas ir pasiruošęs priimti naujus garso duomenis.



4 pav. BSP_AUDIO_OUT_TransferComplete_CallBack funkcijos algoritmas

Išbandžius programa mikrofonas veikė labai gerai ir buvo girdimi įvairūs garsai iš laboratorijos nuo kitų kabinetų durų atsidarymo, peles paspaudimo kolegų kalbėjimo ar net klaviatūros rašymo. Tačiau girdisi lempų ar ventiliatorių užimąs kai niekas nešneka – tai reikėtų pašalinti naudojant filtrus.

2. Pritaikytas audio efektas

Programoje buvo įtrauktas distortion efektas analizuotas trečiame laboratoriniame darbe. Klausant signalų didesni girdimas triukšmas nei programoje be efekto. Taip pat girdisi toks tarsi spragsėjimas esant tylai – lyg laužo degimo garsas.

1 lentelė. Distortion audio efekto pritaikymas

```
/*** Audio data output ***/
void BSP_AUDIO_OUT_HalfTransfer_CallBack(void)
{
  for (int i = 0; i < OUTPUT_BUFFER_SIZE/2; i++) {
    OutputBuffer[i] = Distortion(OutputBuffer[i]);
  }
}

/*** Back to Buffer beginning ***/
void BSP_AUDIO_OUT_TransferComplete_CallBack(void)
{
  BSP_AUDIO_OUT_ChangeBuffer((uint16_t*)&OutputBuffer[0], OUTPUT_BUFFER_SIZE);
  HAL_GPIO_TogglePin(LD3_GPIO_Port, LD3_Pin);

  for (int i = OUTPUT_BUFFER_SIZE/2; i < OUTPUT_BUFFER_SIZE; i++) {
    OutputBuffer[i] = Distortion(OutputBuffer[i]);
  }
}
```

Išvados

1. Laboratorinio darbo metu buvo išanalizuota pateikta programa ir pateikti programos algoritmai tekstine bei grafine forma.
2. Prie projekto garso išvesties buvo pridėtas distortion efektas.