University *of Ljubljana*

Vizualizator frekvenčnega spektra

VIN Projekt

Date: 18/09/2024

Študent

Irinej Slapal

Profesor

Viš. Pred. Dr. Robert Rozman

Univerza v Ljubljani
Fakulteta za Računalništvo in Informatiko

Contents

1 Uvod

V sklopu VIN projekta, sem izdelal prikazovalnik frekvenčnega spektra na STM32H750 vgrajenem sistemu. Sistem pridobiva zvočne podatke preko MEMS digitalnega mikrofona, vgrajenega na plošči in uporablja (CMSIS-DSP) knjižnico za analizo frekvenčnega spektra. Na koncu rezultat analize prikazuje na zaslonu v formatu histograma. Projekt je vključeval štiri glavne faze: usposabljanje mikrofona z BSP knjižnico, verifikacija delovanja mikrofona, vključitev CMSIS-DSP knjižnice in usposabljanje zaslona za prikazovanje z BSP knjižnico.

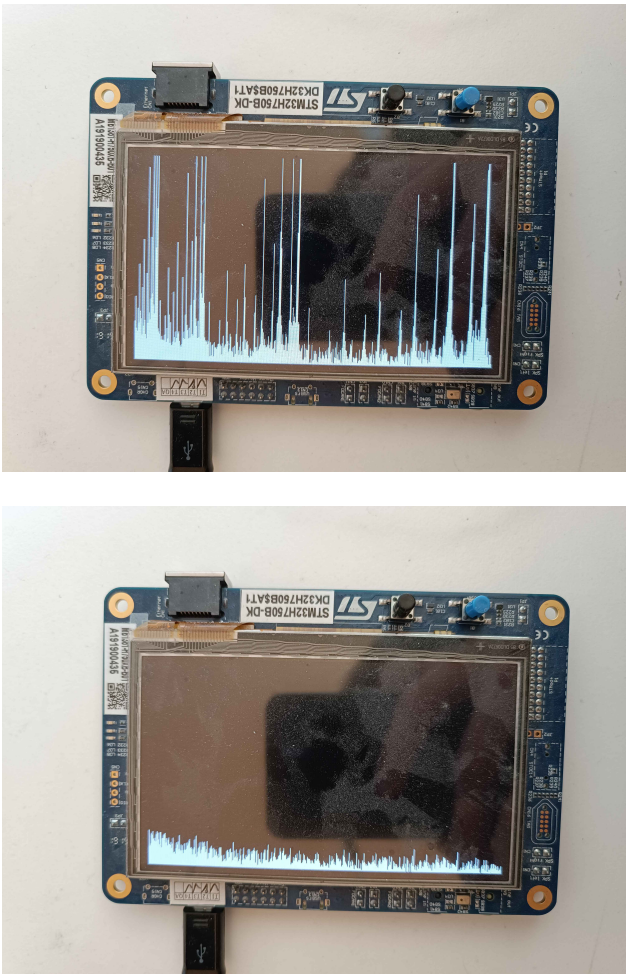

Figure 1: Whistling and no whistling

2 Cilji:

- Pridobiti in obdelati zvočne podatke z mikrofonom.
- Uporabiti DSP knjižnice za izvedbo frekvenčne analize.
- Prikazati frekvenčni spekter na zaslonu v realnem času.

3 Uporabljene komponente:

- stm32h750-bdk plošča vključno z mikrofonom.
- CMSIS-DSP knjižnica.
- stm32h750-BSP knjižnica.
- CUBE-IDE razvojno okolje.

4 Postopek usposabljanja mikrofona:

Pri projektu sem uporabili digitalni MEMS mikrofon, ki ga upravlja SAI4 modul na STM32H750 mikrokontrolerju. Mikrofon vrača signal v obliki PDM (Pulse Density Modulation), ki sem ga moral ustrezno pretvoriti v PCM (Pulse Code Modulation) format za nadaljnjo analizo. Za upravljanje mikrofona in obdelavo signala sem uporabil BSP knjižnico, ki vključuje podporo za WM8994 avdio kodek.

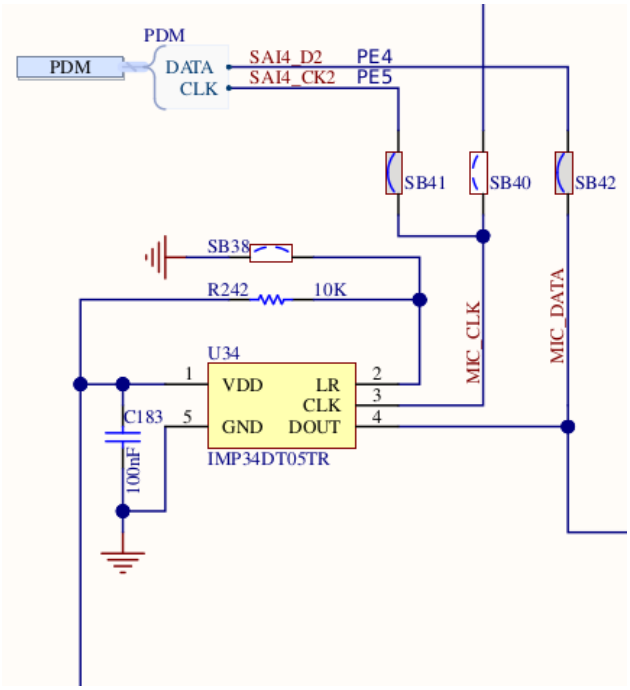

Figure 2: Shema za signale mirkofona

Postopek usposabljanja mikrofona se je začel s konfiguracijo SAI4 modula, ki deluje kot vmesnik za zajemanje PDM signala iz digitalnega MEMS mikrofona. SAI4 modul je bil nastavljen za delovanje v načinu "master receiver", kar omogoča vzorčenje PDM podatkov pri visokih frekvencah. Z uporabo BSP knjižnice in integriranega WM8994 dekoderja sem zajete PDM podatke neposredno bral v SAI4-rx register. WM8994 kodek je bil konfiguriran za delovanje z digitalnim mikrofonom, kar je poenostavilo povezovanje in omogočilo stabilno vzorčenje signalov.

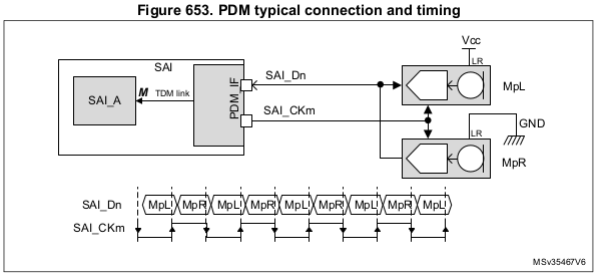

Figure 3: Shema pdm signala

Ko so bili PDM podatki uspešno prebrani v register, sem jih z uporabo PDM2PCM knjižnice pretvoril v PCM format. Ta korak je bil ključen, saj PDM signal ni primeren za neposredno frekvenčno analizo. PCM podatki so se nato shranili v buffer velikosti 2048*16bitov. Za FFT (fast fourier transform) analizo sem jih nato normaliziral.

5 Postopek verifikacije signala iz mikrofona:

Ta korak se je v bistvu zgodil po integraciji DSP knjižnice, saj mi je FFT vrnil čudne vrednosti. To sem storil z uporabo Serial Wire Viewer orodja v CUBE-IDE razvojnem okolju.

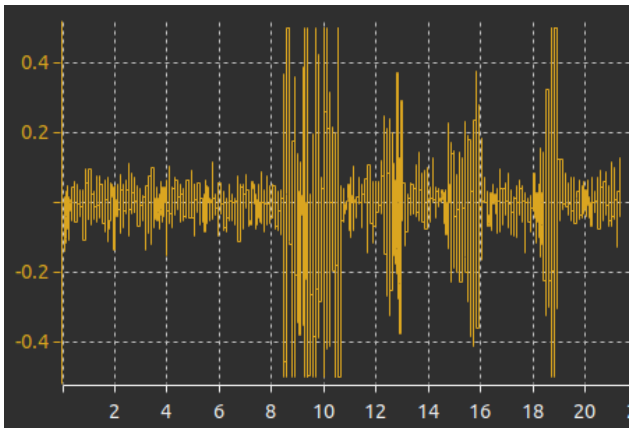

Figure 4: Vizualizacija normaliziranih PCM vrednosti z uporabo SWV orodja

6 Postopek dodajanja CMSIS-DSP knjižnice:

Za dodajanje CMSIS-DSP knjižnice v projekt sem najprej prenesel CMSIS izvorno kodo iz github strani [1]. Nato sem v projektu dodal ustrezne datoteke in poti do knjižnic. Z vključitvijo datotek glave (arm_math.h) v izvorno kodo sem pridobil dostop do različnih DSP funkcij, kot so FFT transformacije, filtriranje in ostale kompleksne matematične operacije.

Med razvojem sem tukaj naletel na težavo, ker je povezovalnik (linker) prenesel vse lookup tabele za FFT in ostale operacije v flash pomnilnik, namesto samo tistih, ki jih je program dejansko potreboval. To je povzročilo preliv pomnilnika in nedelovanje programa. Kasneje sem odkril, da so pri ARM-u to rešili tako, da so implementirali specifične funkcije za vsako dolžino FFT medpomnilnika.

Koda za postopek pretvorbe PCM vrednosti v vrednosti magnitud, iz katerih se lahko potem izračuna frekvenčni spekter:

```
#if defined ( __GNUC__ )  /* !< GNU Compiler */
  ALIGN_32BYTES (uint16_t recordPDMBuf[AUDIO_IN_PDM_BUFFER_SIZE]) __attribute__((section(".
    RAM_D3")));
#endif
ALIGN_32BYTES (uint16_t  RecPlayback[AUDIO_BUFF_SIZE]);
HAL_SAI_Receive(&haudio_in_sai, (uint8_t *) &recordPDMBuf, AUDIO_IN_PDM_BUFFER_SIZE, 10);
BSP_AUDIO_IN_PDMToPCM(1, (uint16_t*)&recordPDMBuf[0], (uint16_t *) &RecPlayback[playbackPtr
    ]);

arm_rfft_fast_instance_f32 fft_instance;
arm_rfft_fast_init_1024_f32(&fft_instance);
playbackPtr += AUDIO_IN_PDM_BUFFER_SIZE/sizeof(uint16_t)/4/2;
if(playbackPtr >= AUDIO_BUFF_SIZE)
{
    playbackPtr = 0;
    for (int i = 0; i < FFT_LENGTH; i++)
    {
        input_fft[i] = ((float32_t)((int16_t)
        RecPlayback[i*2]))/65536.0f;//*window[i];
    }
    arm_rfft_fast_f32(&fft_instance, input_fft, output_fft, 0);
    arm_cmplx_mag_f32(output_fft, output_fft_mag, FFT_LENGTH/2);
}
```

Za izračun frekvence f iz magnitudne vrednosti v spektru lahko uporabimo naslednjo enačbo:

$$f = \frac{n \cdot f_s}{N} \tag{1}$$

kjer je:

- n - indeks frekvenčnega komponenta v FFT rezultatu (iteracija po medpomnilniku),
- f_s - frekvenca vzorčenja (sampling rate),
- N - število točk v FFT (velikost FFT medpomnilnika).

7 Postopek usposabljanja zaslona:

Za usposabljanje zaslona, sem vzel prej izdelan projekt kot primer za uporabo BSP knjižnice za uporabo LCD zaslona z gihtub strani [2], ki vključuje podporo za specifičen uporabljen mikrorkmilnik. Nakoncu sem po izvedbi FFT transformacije iteriral čez array magnitud in prikazal frekvenčni spekter, z risanjem stolpcev, ki predstavljajo amplitudo posameznih frekvenc.

8 Zaključek:

Začetni cilj za projekt je bila izdelava uglaševalca kitare. Vendar sem veliko časa porabil za iskanje informacij po dokumentaciji, saj nisem našel nobenih primerov za uporabo mikrofona. Na koncu sem bil premalo vztrajen in mi je zmanjkalo časa, da bi se lahko naučil dovolj o digitalnem procesiranju signalov in kar sem si na začektu zastavil. Potem sem to kar sem imel le na grobo povezal v neko celoto.
Večino informacij glede mikrofona in pdm,pcm pretvorbe sem dobil iz uradne dokumentacije [3], in poleg tega tudi postal boljši pri branju in iskanju informacij. V prihodnosti bom projekt dokončal v končni izdelek uglaševalca kitare.

References

[1] ARM-software. Cmsis-dsp library, 2024.

[2] Aljaz Justin. Display totorial, 2024.

[3] STM32. Datasheets, 2024.