Desevolvimento de um software para controle de rádio FM no dispositivo GCW

Aluno: Marcos Paulo de Souza

Orientador: Mauro Marcelo Mattos

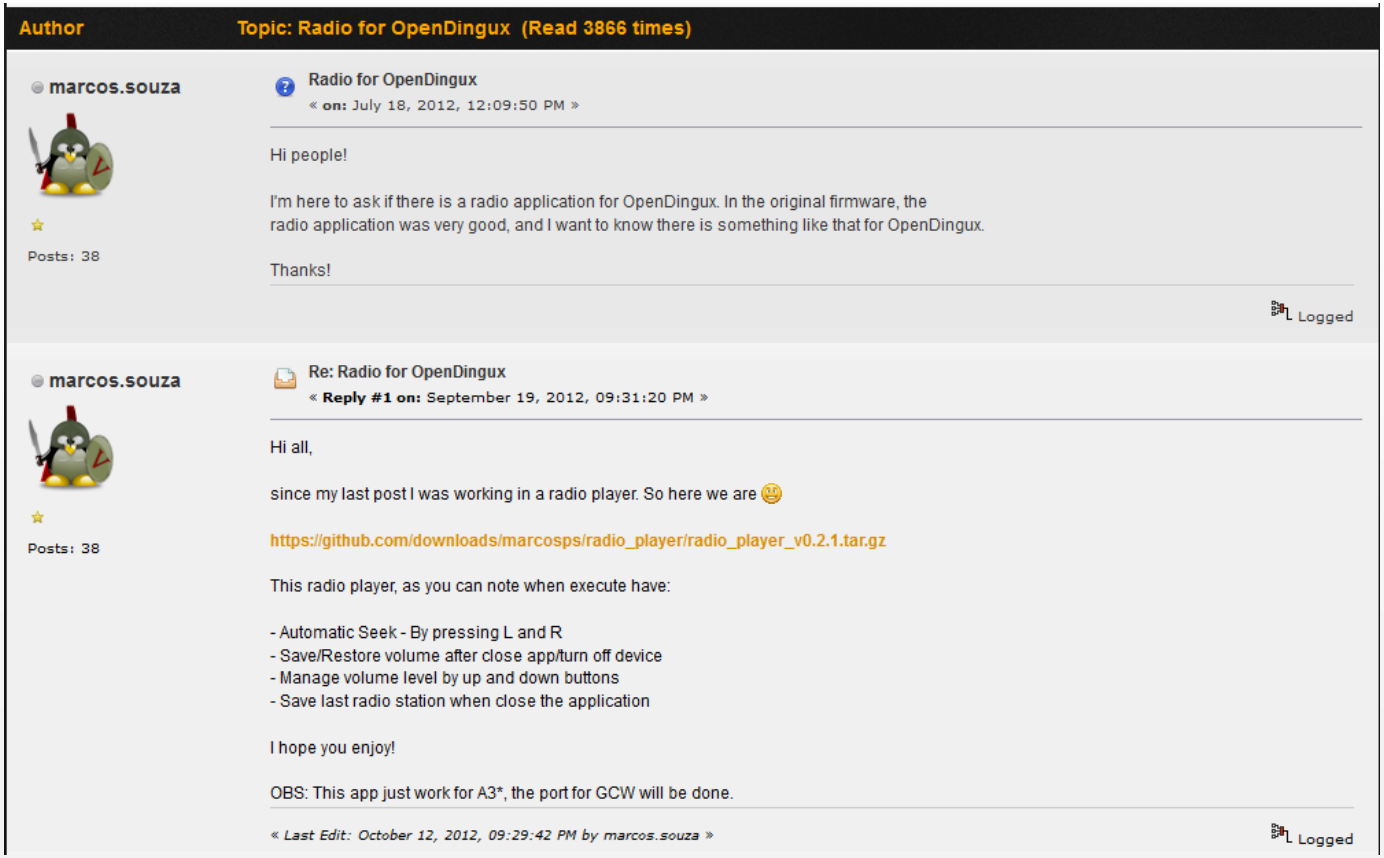
FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Funcamentação teórica
- Desenvolvimento
- Implementação
- Conclusão
- Extensões

Introdução

Motivação:



Author | **Topic: Radio for OpenDingux (Read 3866 times)**

marcos.souza

Posts: 38

Radio for OpenDingux
« **on:** July 18, 2012, 12:09:50 PM »

Hi people!

I'm here to ask if there is a radio application for OpenDingux. In the original firmware, the radio application was very good, and I want to know there is something like that for OpenDingux.

Thanks!

Logged

marcos.souza

Posts: 38

Re: Radio for OpenDingux
« **Reply #1 on:** September 19, 2012, 09:31:20 PM »

Hi all,

since my last post I was working in a radio player. So here we are

https://github.com/downloads/marcosps/radio_player/radio_player_v0.2.1.tar.gz

This radio player, as you can note when execute have:

- Automatic Seek - By pressing L and R
- Save/Restore volume after close app/turn off device
- Manage volume level by up and down buttons
- Save last radio station when close the application

I hope you enjoy!

OBS: This app just work for A3*, the port for GCW will be done.

« Last Edit: October 12, 2012, 09:29:42 PM by marcos.souza »

Logged

Introdução

- Como surgiu?

- Solução

Objetivos

- Corrigir o device driver do microchip de rádio para que tenha as funcionalidades básicas do hardware RDA5807.
- Disponibilizar um aplicativo que interage com este device driver e que permita ao usuário ouvir estações de rádio no dispositivo GCW.

Fundamentação Teórica

- Sistema operacional Linux
- APIs *ALSA*, *SDL, Video4Linux*
- Chamada de device drivers
- Plataforma GCW
- *Cross Compiler*

Trabalhos Correlatos

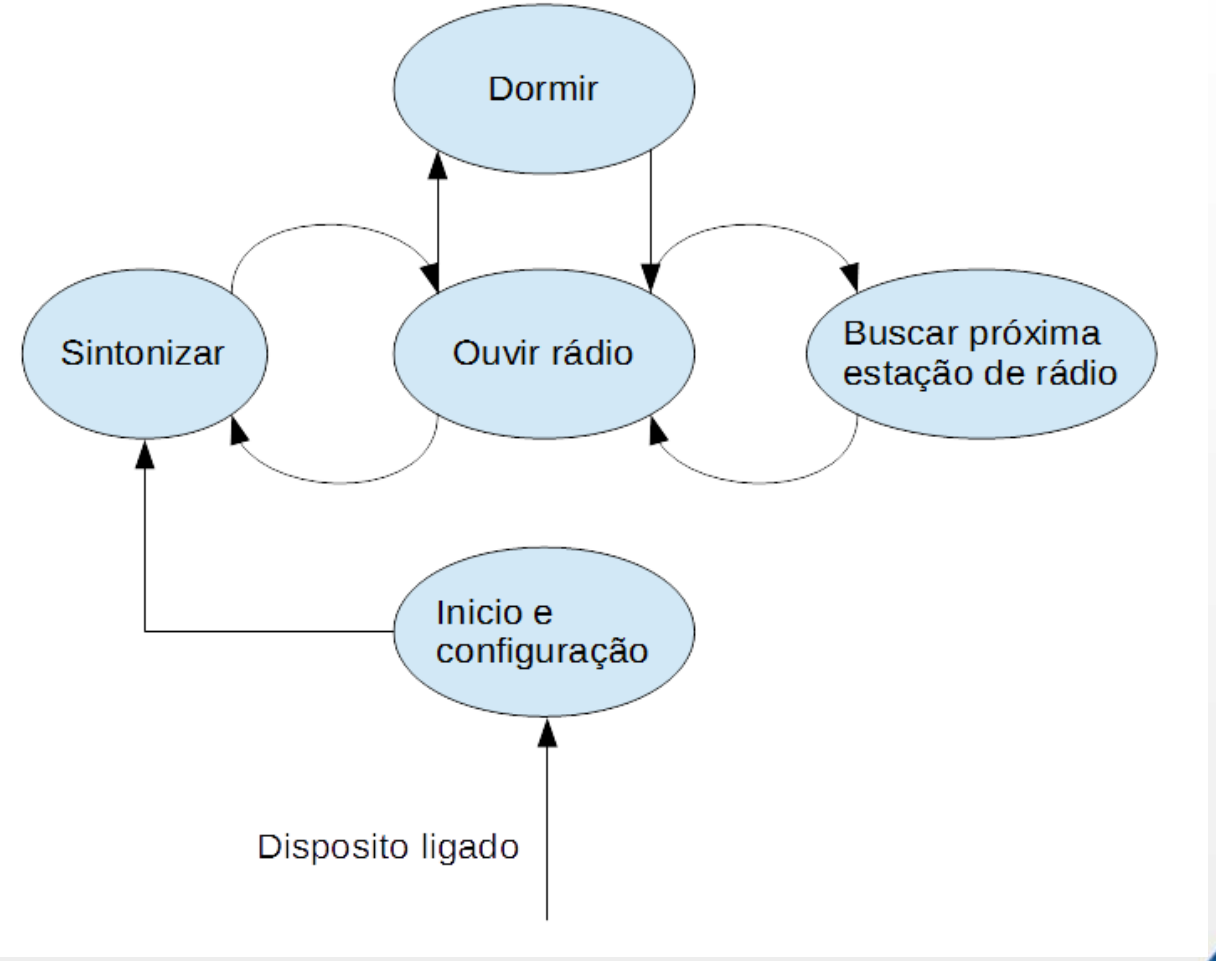
- Minivosc
- Toolchain para Linux embarcado

Requisitos

- Permitir ouvir estações de rádio no dispositivo GCW (RF);
- Permitir cadastrar as estações de rádio favoritas (RF);
- Permitir ouvir as estações de rádio utilizando fones de ouvido e alto-falantes (RF);
- Guardar e restaurar as últimas opções do usuário quando o aplicativo ser reiniciado (RNF);
- Utilizar a biblioteca SDL para controle da interface com o usuário (RNF);
- Utilizar a linguagem C para desenvolver o aplicativo (RNF);
- Utilizar o *framework ALSA* para interagir com o controle de áudio do dispositivo (RNF);

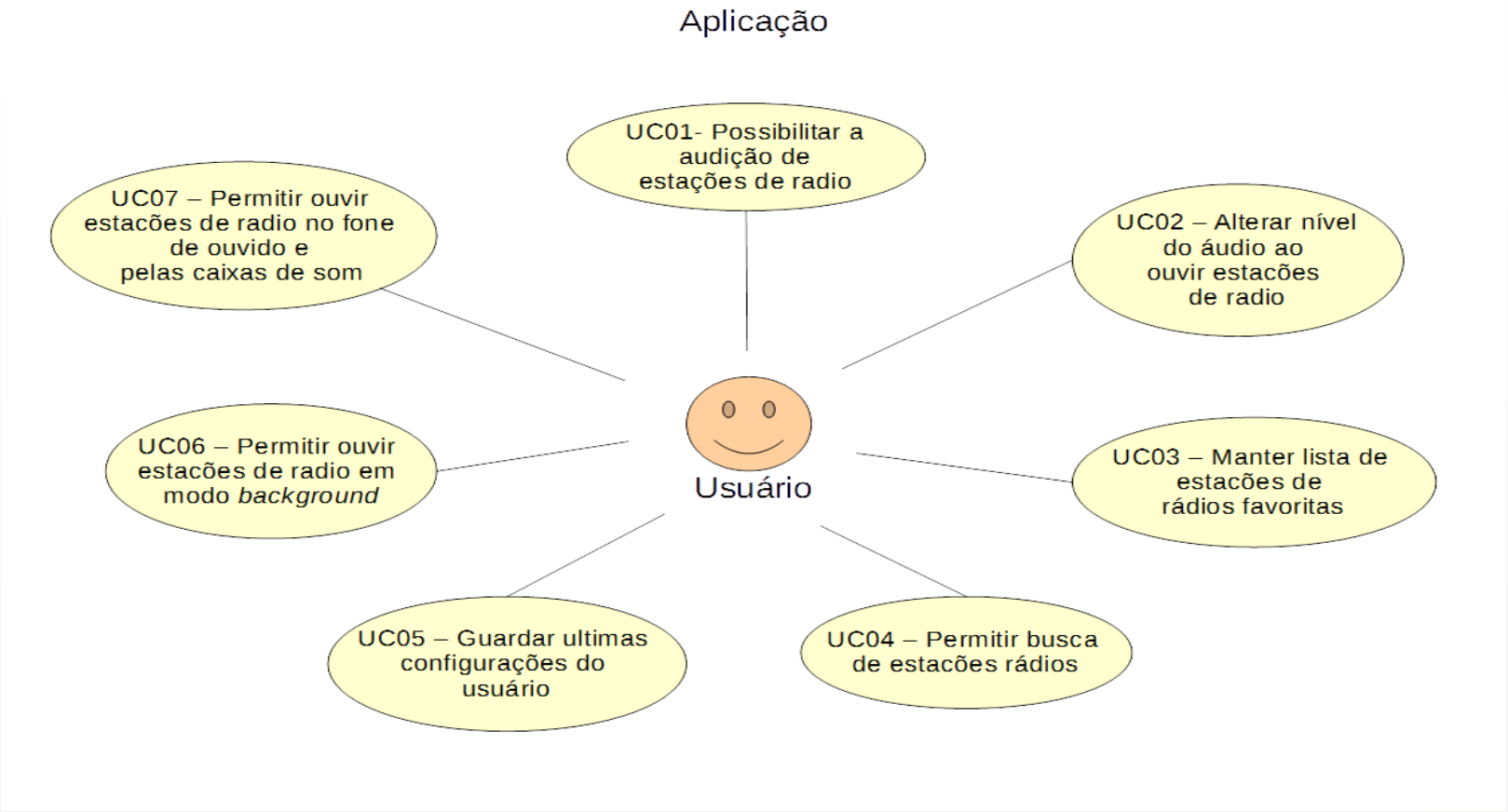
Especificação

Diagrama de estados do chip de rádio RDA5807



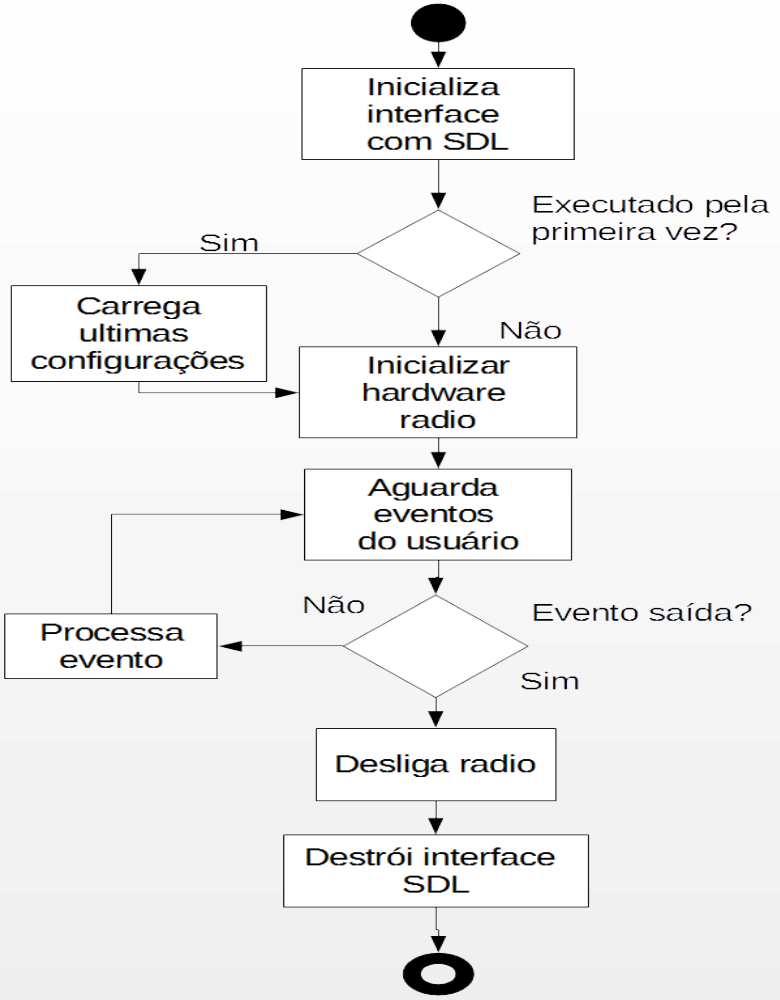
Especificação

Diagrama de casos de uso da aplicação



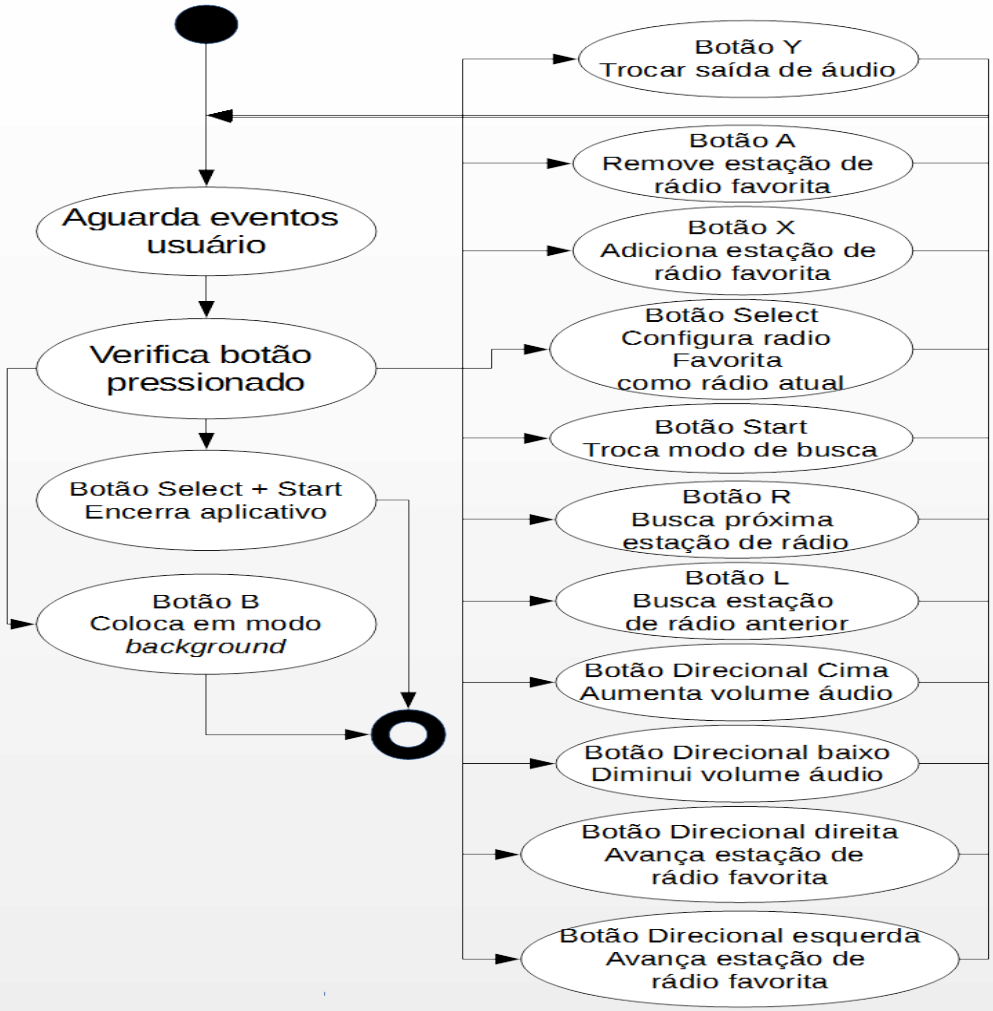
Especificação

Diagrama de atividades do aplicativo



Especificação

Diagrama de estados do aplicativo



Implementação

- Kernel
  - Device driver
- Aplicativo
  - Tela
  - Dados
  - Configurações rádio
  - Configurações audio

Implementação – Device Driver

Código que busca a próxima estação de rádio.

```
static int rda5807_seek_frequency(struct rda5807_driver *radio,
                                  int upward, int wrap)
{
        u16 mask = 0;
        u16 val = 0;
        int ret, count = 0;

        /* TODO: Seek threshold is configurable. How should the driver handle
         *       this configuration?
         */
        /* seek up or down? */
        mask |= RDA5807_MASK_CTRL_SEEKUP;
        if (upward)
                val |= RDA5807_MASK_CTRL_SEEKUP;
        /* wrap around at band limit? */
        mask |= RDA5807_MASK_CTRL_SKMODE;
        if (!wrap)
                val |= RDA5807_MASK_CTRL_SKMODE;
        /* seek command */
        mask |= RDA5807_MASK_CTRL_SEEK;
        val  |= RDA5807_MASK_CTRL_SEEK;

        return rda5807_update_reg(radio, RDA5807_REG_CTRL, mask, val);
}
```

Implementação – Device Driver

Código ajustado que busca a próxima estação de rádio.

```
static int rda5807_seek_frequency(struct rda5807_driver *radio,
                                  int upward, int wrap)
{
        u16 mask = 0;
        u16 val = 0;
        int ret, count = 0;

        /* TODO: Seek threshold is configurable. How should the driver handle
         *       this configuration?
         */
        /* seek up or down? */
        mask |= RDA5807_MASK_CTRL_SEEKUP;
        if (upward)
                val |= RDA5807_MASK_CTRL_SEEKUP;
        /* wrap around at band limit? */
        mask |= RDA5807_MASK_CTRL_SKMODE;
        if (!wrap)
                val |= RDA5807_MASK_CTRL_SKMODE;
        /* seek command */
        mask |= RDA5807_MASK_CTRL_SEEK;
        val  |= RDA5807_MASK_CTRL_SEEK;

        ret = rda5807_update_reg(radio, RDA5807_REG_CTRL, mask, val);
        if (ret < 0)
                return ret;

        while (1) {
                /*
                 * The programming guide says we should wait for 35 ms for each
                 * frequency tested.
                 */
                msleep(35);

                ret = rda5807_i2c_read(radio->i2c_client,
                                       RDA5807_REG_SEEK_RESULT);
                if (ret < 0)
                        return ret;

                /* Seek done? */
                if (ret & RDA5807_MASK_SEEKRES_COMPLETE)
                        return 0;

                /*
                 * Channel spacing is 100 kHz.
                 * TODO: Should we support configurable spacing?
                 */
                count++;
                if (count > (RDA5807_FREQ_MAX_KHZ - RDA5807_FREQ_MIN_KHZ) / 100)
                        return -ETIMEDOUT;
        }
}
```

Implementação - Aplicativo

Configurações de chip de rádio RDA5807

```
/* frequency in MHz */
void setup(float frequency)
{
        control.id = V4L2_CID_AUDIO_MUTE;
        control.value = 0;

        if (ioctl(fd, VIDIOC_S_CTRL, &control) < 0) {
                perror("ioctl: set: mute off");
                fprintf(stderr, "We can't continue without turns mute to off. Aborting.\n");
                exit(1);
        }

        if (ioctl(fd, VIDIOC_G_TUNER, &tuner) < 0) {
                perror("ioctl: set: get tuner");
                fprintf(stderr, "We can't continue without a tuner. Aborting.\n");
                exit(1);
        }

        set_frequency(frequency);

        control.id = V4L2_CID_AUDIO_VOLUME;
        control.value = 15;

        if (ioctl(fd, VIDIOC_S_CTRL, &control) < 0) {
                perror("ioctl: set volume");
                fprintf(stderr, "Using the default volume level.\n");
        }
}
```

Implementação - Aplicativo

Configurações de audio do dispositivo GCW

```
/* Controls the alsamixer atributes of GCW device */
void mixer_control(int mode, long *volume, long *min, long *max)
{
        snd_mixer_t *handle;
        snd_mixer_selem_id_t *sid;
        snd_mixer_elem_t *elem;

        snd_mixer_selem_channel_id_t channel = SND_MIXER_SCHN_FRONT_LEFT;

        snd_mixer_open(&handle, 0);
        snd_mixer_attach(handle, "default");
        snd_mixer_selem_register(handle, NULL, NULL);
        snd_mixer_load(handle);

        snd_mixer_selem_id_alloca(&sid);
        snd_mixer_selem_id_set_index(sid, 0);

        if (mode == VOLUME_GET || mode == VOLUME_SET) {
                snd_mixer_selem_id_set_name(sid, "Headphone");
                elem = snd_mixer_find_selem(handle, sid);

                if (mode == VOLUME_GET) {
                        snd_mixer_selem_get_playback_volume(elem, channel, volume);
                        snd_mixer_selem_get_playback_volume_range(elem, min, max);
                } else if (mode == VOLUME_SET) {
                        printf("GCW: Volume set to %ld\n", *volume);
                        snd_mixer_selem_set_playback_volume_all(elem, *volume);
                }

                /* adjust volume to Bypass too */
                snd_mixer_selem_id_set_name(sid, "Line In Bypass");
                elem = snd_mixer_find_selem(handle, sid);

                if (mode == VOLUME_SET) {
                        printf("Line in Bypass: Volume set to %ld\n", *volume);
                        snd_mixer_selem_set_playback_volume_all(elem, *volume);
                }

        } else if (mode == HEADPHONE_TURN_ON || mode == HEADPHONE_TURN_OFF) {
                snd_mixer_selem_id_set_name(sid, "Headphone Source");
                elem = snd_mixer_find_selem(handle, sid);

                if (mode == HEADPHONE_TURN_ON) {
                        printf("Headphone Source: Line In\n");
                        snd_mixer_selem_set_enum_item(elem, channel, 1);
                } else if (mode == HEADPHONE_TURN_OFF) {
                        printf("Headphone Source: PCM\n");
```

Implementação - Aplicativo

Desenhar na tela as rádios favoritas

```
/* To be able to draw borders, we need to draw a bigger rect, and after a small one
 * filled by black color */
static void draw_favrads_rects()
{
        Uint32 black_color = SDL_MapRGB(screen->format, 0, 0, 0);
        Uint32 green_color = SDL_MapRGB(screen->format, 0, 255, 0);
        Uint32 white_color = SDL_MapRGB(screen->format, 255, 255, 255);

        int i = 0;

        for (i = 0; i < 5; i++) {
                /* Selected favorite radio has green border */
                if (i == curr_fav)
                        SDL_FillRect(screen, &favrad_rects[i], green_color);
                else
                        SDL_FillRect(screen, &favrad_rects[i], white_color);

                SDL_FillRect(screen, &favrad_rects_border[i], black_color);

                printf("Radio %s\n", favrads.radio[i]);

                int freq_size = strlen(favrads.radio[i]);

                char freq[6];

                if (freq_size == 1)
                        strcpy(freq, "-");
                else
                        strcpy(freq, favrads.radio[i]);

                /* Draw favorite radio into rect */
                desc_fav_rad_info = TTF_RenderText_Solid(desc_fav_rad_font, freq, font_color);
                apply_surface(favrad_rects[i].x + 10, 35, desc_fav_rad_info, screen);
        }
        SDL_Flip(screen);
}
```

Implementação - Aplicativo

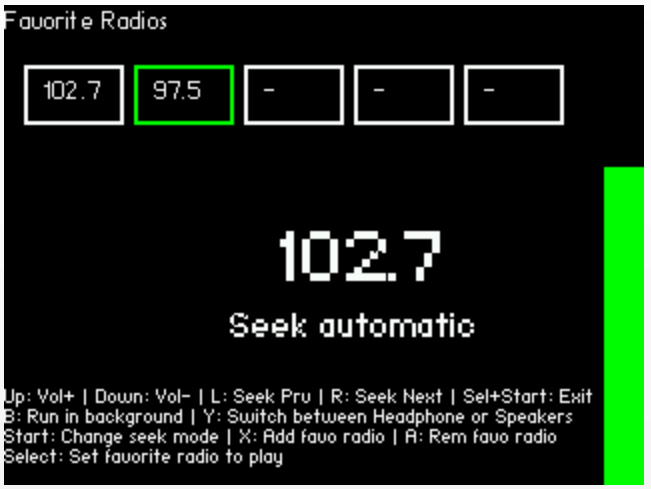
Guardar as configurações do usuario

```
/* save/restore sound level */
void handle_sound_level(int mode, long *volume)
{
        char line[3];

        if (path[0] != '0') {
                sprintf(aux_path, "%s/%s", path, "last_volume");

                if (mode == FILE_VOLUME_READ) {
                        file = fopen(aux_path, "r");

                        /* let the volume parameter as it was */
                        if (!file)
                                return;

                        fgets(line, 3, file);
                        sscanf(line, "%ld", volume);

                } else if (mode == FILE_VOLUME_WRITE) {
                        file = fopen(aux_path, "w");

                        if (!file) {
                                fprintf(stderr, "Cannot set the actual volume level!\n");
                                return;
                        }

                        fprintf(file, "%ld", *volume);
                }
                if (file)
                        fclose(file);
        }
}
```

Operacionalidade da Implementação

Aplicativo fazendo busca de estações de rádio

Aplicativo adicionou rádios favoritas



Resultados e Discussões

Função	Aplicativo rádio	Minivosc	Toolkit
Implementação em *device driver*	X	X	
Implementação na camada do usuário	X		X
Ambiente Linux	X	X	X
Dispositivos embarcados	X		X

Conclusões e Sugestões

- Foi possivel estudar a estrutura de device drivers do Linux, como interagir com device drivers utilizando a linguagem C e como interagir com o ALSA
- Os objetivos foram atingidos
  - Disponibilizar uma aplicação para permitir ao usuário operar o rádio FM do dispositivo;
  - Validar o uso da ferramenta através de um estudo de caso envolvendo operações de uso no dispositivo real

Conclusões e Sugestões

Melhorias podem ser introduzidas neste trabalho, tais como:

- melhorar a interface do usuário, deixando menos parecido com um terminal Linux;
- utilizar um banco de dados como o SQLite para armazenar as preferência de usuário ao invés de utilizar arquivos com texto;
- permitir uma configuração de nível de sinal aceito pelo rádio, para possibilitar o usuário ouvir estações de rádio de uma maior distância;
- submeter o *driver* de dispositivo utilizado neste trabalho para a versão oficial do Linux;