ULTRON - CENTRAL MULTIMÍDIA
COM RECONHECIMENTO DE COMANDOS DE VOZ

Daniel Borges Pinheiro – 12/0114283

Graduação em Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
E-mail: danborges06@hotmail.com.br

David da Silva Ferreira - 14/0018913

Graduação em Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
E-mail: dsf.604@gmail.com

1. JUSTIFICATIVA

A constante evolução tecnológica na indústria automobilística vem promovendo a inserção de mecanismos e dispositivos eletrônicos capazes de proporcionar maior comodidade e segurança ao usuário. Dentre os dispositivos que trouxeram mudanças significativas a este setor, pode-se destacar a central multimídia.

O termo multimídia, como o próprio nome diz, se refere à capacidade de comportar diversas funcionalidades em termos de meios de comunicação. Portanto, a central trata-se de um sistema capaz de unir e gerenciar o funcionamento destes recursos. Dentre as principais funcionalidades observadas em aparelhos como este, estão: player de som e vídeo, rádio, leitor de dispositivos USB, interação com dispositivos celulares, entre outras.

Tendo em vista que atualmente quase todo carro possui um tocador de CD ou mesmo DVD player, está claro que equipamentos automotivos de multimídia são uma forte demanda no mercado. No entanto, pelo fato de mídias físicas estão cada vez mais em desuso, dando lugar ao armazenamento USB ou mesmo on-line, foi assumindo-se que é suficiente lançar um equipamento mais enxuto neste quesito. Porém tendo somente esta funcionalidade de multimídia, é possível que não seja um chamariz suficiente para o consumidor e também não justificaria a escolha desta proposta como trabalho.

A partir de uma pesquisa a respeito das necessidades observadas em sistemas como esse, observou-se que o fato de seu controle ser manual faz com que o motorista desvie sua atenção do trânsito. Desvios de atenção como esse podem ocasionar diversos acidentes de trânsito, colocando vidas em riscos. Tendo isso em vista, procurou-se idealizar um sistema que pudesse facilitar a sua interação com o usuário, de modo que este precise desviar o mínimo de atenção para controlá-lo.

A proposta do projeto é a implementação de uma central multimídia, que seja capaz de realizar algumas das principais funcionalidades apresentadas, tais como a reprodução de arquivos mediante inserção de dispositivo USB. O sistema proposto opera mediante o reconhecimento de comandos de voz, de modo a garantir que o usuário não precise retirar suas mãos do volante ou direcionar sua visão para a interface da central.

2. OBJETIVOS

O projeto consiste no desenvolvimento de uma central multimídia para automóveis. Esta central comportará uma interface visual com o usuário, permitindo que este seja capaz de acessar com maior facilidade os seus recursos. O sistema apresentará como principal funcionalidade o uso de um reconhecedor de comandos de voz, de modo a executar funções referentes à reprodução de arquivos de áudio e vídeo. O acesso aos arquivos se dará a partir da conexão de um dispositivo USB. O sistema está aberto a possíveis acréscimos de funcionalidades, tanto em software como hardware.

3. REQUISITOS

O sistema será implementado em um Raspberry Pi, visto que o seu processador é capaz de atender aos requisitos do projeto, conciliado a um baixo consumo de energia e às suas interfaces USB e HDMI.

Em termos de hardware, apenas os modelos de maior desempenho da placa, como o Raspberry Pi 3, são capazes de suprir as necessidades de alta capacidade de processamento do produto. No entanto, para que possa haver interação com usuário e acesso aos recursos multimídia, o veículo do proprietário deverá estar equipado com um sistema de som com entrada auxiliar ou bluetooth e uma tela com entrada HDMI ou vídeo composto. Caso sejam incluídas outras funcionalidades, serão necessários outros módulos ou sensores, que deverão estar devidamente instalados no produto ou veículo.

Como o sistema funcionará baseado no reconhecimento de voz e o Raspberry não apresenta entrada de áudio, será necessário o uso de um microfone que funcione via USB.

Já em termos de software, será necessário a adaptação de uma distribuição Linux específica mais enxuta, como a LibreELEC, que faça uso da aplicação de interface multimídia e do player Kodi. Para o uso do comando de voz, serão utilizados processos rodando em segundo plano. Estes processos utilizarão a biblioteca PocketSphinx, que realiza o reconhecimento da voz e retorna uma String com o que foi interpretado. [1]

4. BENEFÍCIOS

Além da comodidade, dados os recursos de interface visual e reprodução de arquivos, a central Ultron oferecerá mais segurança ao proprietário, visto que proporcionará ao usuário um controle do sistema sem que este precise desviar a atenção ao trânsito. Por isto, foi pensado na funcionalidade de controle por comandos de voz, pois além de agregar bastante ao produto, também atende a outra demanda que vem crescendo cada vez mais no ramo automobilístico, que é o da acessibilidade e segurança automotiva.

5. MATERIAIS UTILIZADOS

- 01 Raspberry Pi 3
- 01 Microfone – saída USB
- 01 Monitor com entrada HDMI
- 01 Pen drive

6. FUNCIONAMENTO DO SISTEMA

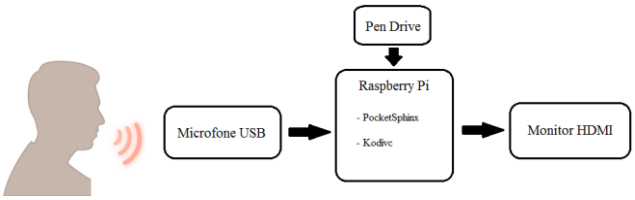

Figura 1: Diagrama geral do sistema

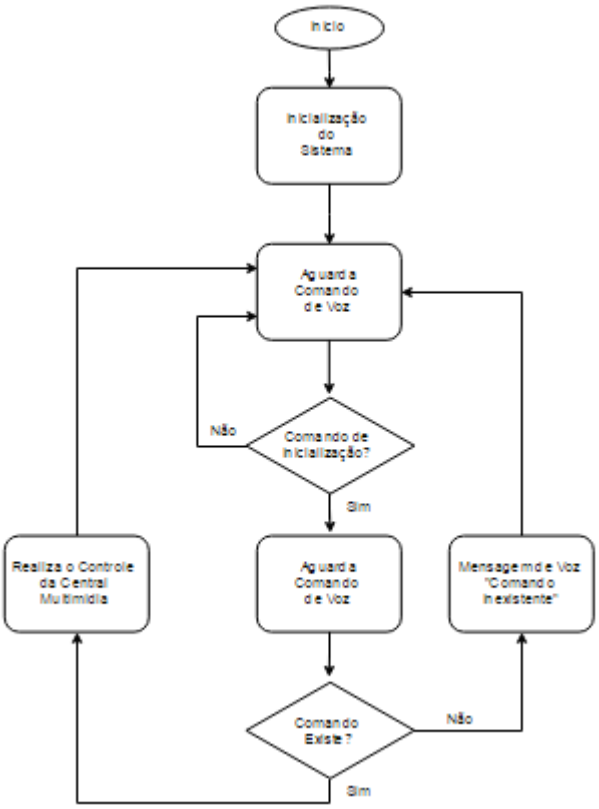

Figura 2: Fluxograma do sistema

A comunicação pela qual se dará o controle do aplicativo Kodi será via protocolo HTTP, através de comandos que utilizam a interface JSON-RTP [3]. Para isto, os comandos são enviados pelo método de transporte POST. Este método de transporte foi implementado via comando CURL. Basicamente, o que o sistema faz é receber o comando de voz através da biblioteca que roda em segundo plano, verificar se é um comando válido entre os que foram estipulados no sistema, construir uma String que representa a requisição HTTP daquele comando específico e executá-la via comando system. A requisição HTTP via comando CURL é executado da seguinte forma:

curl -X POST -H \"Content-Type: application/json\" -d '<command_ultron>' localhost:8080/jsonrpc

O campo <command_ultron> se trata do comando a ser executado, no formato JSON. Um exemplo de comando no formato adequado que pode ser utilizado é [4]:

{"jsonrpc":"2.0","method":"Player.PlayPause", "params": {"playerid":0},"id":1}

Este comando, em específico, pausa ou toca o que estiver sendo executado no player. A lista com os demais comandos JSON utilizados constam nos anexos.

O sistema possui 16 comandos básicos, sendo eles os seguintes: ultron, home, up, down, left, right, select, back, play, pause, stop, next, previous, incremente, decrement, mute, maximum, e shutdown.

- Ultron: realiza a chamada do sistema para que ele comece a escutar os comandos de voz.
- Home: direciona a multimídia para a página inicial.
- Up: direciona a seleção do menu do sistema para cima.
- Down: direciona a seleção do menu do sistema para baixo.
- Left: direciona a seleção do menu do sistema para a esquerda, ou abre um sub-menu, quando existente.
- Right: direciona a seleção do menu do Sistema para a direita, ou retorna para o menu anterior, quando existente.
- Select: seleciona a opção marcada no menu.
- Back: retorna para o menu anterior.
- Play: executa a mídia que estiver selecionada.
- Pause: pausa a mídia que estiver sendo executada.
- Stop: para de executar a mídia atual.
- Next: direciona para a próxima mídia.
- Previous: direciona para a mídia que foi executada anteriormente.
- Increment: realiza o aumento do volume do sistema gradativamente à medida que o comando é executado.
- Decrement: realiza a redução do volume do Sistema gradativamente à medida que o comando é executado.
- Mute: seta o volume do sistema para zero.
- Maximum: seta o volume do sistema para o máximo.
- Shutdown: desliga o sistema.

Para que os comandos sejam realizados é necessário que o usuário mande o comando “Ultron”, no qual o sistema entende que está sendo chamado e então aguarda o comando a ser executado. Ao compreender o comando de chamada, o sistema responderá “Hello”. Caso o comando dito não exista, o sistema responderá “Command inexistent” e aguardará ser chamado novamente.

7. DEPENDÊNCIAS E IMPLEMENTAÇÃO

Para que o programa desenvolvido possa funcionar dentro do ambiente Linux, é necessário que sejam instalados alguns módulos e bibliotecas a parte.

Para ser realizado o reconhecimento de voz, é utilizada a biblioteca PocketSphinx, uma

adaptação do projeto CMU Sphinx que é implementada em C e realiza o reconhecimento de fala com um baixo custo computacional, sendo a opção mais apropriada para dispositivos portáteis e está disponível no repositório oficial do Debian. Para que a biblioteca possa funcionar, são requeridos três módulos. Um deles é o modelo acústico de fala em inglês, que é responsável pelo reconhecimento sonoro dos sons de fala, que pode ser baixado no próprio repositório da biblioteca [5]. Os outros módulos são o modelo de linguagem e o dicionário, que possuem os fonemas das palavras reconhecíveis e as palavras e expressões disponíveis, respectivamente. Estes módulos podem ser baixados, mas é recomendável que sejam bem limitados ao que se possui de comando, para melhorar a velocidade e a assertividade do reconhecimento, sendo compiladas por uma ferramenta [6].

Outras dependências são o programa Curl, que envias os comandos ao Kodi por meio de requisições post locais, o módulo Pulse Áudio do Linux e o próprio Kodi, que deve ser configurado para receber comandos JSON-RPC por HTTP. Ambos então disponíveis no repositório oficial do Debian.

8. DISCUSSÕES E CONCLUSÃO

Após instaladas todas as dependências e pacotes necessários para a execução do programa, o sistema apresentou funcionamento satisfatório no PC através da distribuição Ubuntu. Ao realizar o teste na placa Raspberry com a distribuição Raspbian como base, apesar de possuir o mesmo código e não ter apresentado nenhum erro ou warning vindo do compilador, não funcionou corretamente.

Alguns problemas foram apresentados apenas na implementação na Raspberry. O aplicativo curl, quando chamado, não reconhecia a sintaxe do comando empregado via chamada do terminal. O comando aplay do módulo Alsa do Linux não tocava os sons de sinalização de forma adequada, de maneira que não era mais possível se guiar por áudio, o aplicativo Kodi não permitia ser alternado e ser executado em segundo plano. Devido a esses e outros problemas, não foi possível apresentar na placa a execução do sistema e testar integralmente o código desenvolvido.

Muitos destes problemas se devem ao fato deste projeto depender extensivamente de módulos e aplicativos externos, também associado a execução na distribuição Raspibian, que às vezes possuía pacotes menos recentes em relação aos das distribuições para PC.

9. REFERÊNCIAS

[1] PocketSphinx. Disponível em: <https://github.com/cmusphinx/pocketsphinx>

[2] Kodivc. Disponível em: <https://github.com/kempniu/kodivc>

[3] JSON-RPC API. Disponível em: <http://kodi.wiki/?title=JSON-RPC_API>

[4] JSON-RPC API. Disponível em: <http://kodi.wiki/view/JSON-RPC_API/Examples>

[5] JSON-RPC API. Disponível em: https://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/

[6] Building language model. Disponível em: https://cmusphinx.github.io/wiki/tutoriallm/#building-a-simple-language-model-using-web-service

[7] JSON API Examples. Disponível em: <http://habitech.s3.amazonaws.com/PDFs/STR/STREAM_Box%20Note%20-%20JSON%20API%20Examples%20for%20ContentPlayer%20(Isengard).pdf>

10. ANEXOS

10.1. Comandos JSON para o Kodi

home	{"jsonrpc":"2.0","method":"Input.Home"}
up	{"jsonrpc":"2.0","method":"Input.Up"}
down	{"jsonrpc":"2.0","method":"Input.Down"}
left	{"jsonrpc":"2.0","method":"Input.Left"}
right	{"jsonrpc":"2.0","method":"Input.Right"}
select	{"jsonrpc":"2.0","method":"Input.Select"}
back	{"jsonrpc":"2.0","method":"Input.Back"}
Play/Pause	{"jsonrpc": "2.0", "method": "Player.PlayPause", "params": { "playerid": 1 }, "id": 1}
Stop	{"jsonrpc": "2.0", "method": "Player.Stop", "params": { "playerid": 1 }, "id": 1}
Fast Forward	{"jsonrpc": "2.0", "id": 1, "method": "Player.SetSpeed", "params": {"playerid": 1, "speed": "increment"}}
Rewind	{"jsonrpc": "2.0", "id": 1, "method": "Player.SetSpeed", "params": {"playerid": 1, "speed": "decrement"}}
Skip Forward	{"jsonrpc":"2.0","method":"Player.GoTo","id":1,"params":{"playerid":1,"to":"next"}}
Skip Previous	{"jsonrpc":"2.0","method":"Player.GoTo","id":1,"params":{"playerid":1,"to":"previous"}}

Figura 3: Comandos JSON

10.2. Implementação do sistema em linguagem C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <unistd.h>

#define TRUE 1
#define FALSE 0

typedef struct {
    char command[50];
    char json_command[100];
} ultron_command;

//Coloca os caracteres da string em maiúsculo
void StrUp(char *str);

int main()
{
    int i, ultron_called, command_found;
    char call_ultron[] = "pocketsphinx_continuous -inmic yes -lm
%HOME/ultron/model/lm/langmodel.lm -dict %HOME/ultron/model/lm/dictionary.dic -hmm
%HOME/ultron/model/en-us";
    char buffer[500];
    char command[100], curl_command[500];
    ultron_command all_commands[16];

    strcpy(all_commands[0].command, "HOME");
    strcpy(all_commands[0].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Input.Home\"}");
    strcpy(all_commands[1].command, "UP");
    strcpy(all_commands[1].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Input.Up\"}");
    strcpy(all_commands[2].command, "DOWN");
    strcpy(all_commands[2].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Input.Down\"}");
    strcpy(all_commands[3].command, "LEFT");
```

```
    strcpy(all_commands[3].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Input.Left\"}");
    strcpy(all_commands[4].command, "RIGHT");
    strcpy(all_commands[4].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Input.Right\"}");
    strcpy(all_commands[5].command, "SELECT");
    strcpy(all_commands[5].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Input.Select\"}");
    strcpy(all_commands[6].command, "BACK");
    strcpy(all_commands[6].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Input.Back\"}");
    strcpy(all_commands[7].command, "PLAY");
    strcpy(all_commands[7].json_command, "{\"jsonrpc\": \"2.0\", \"method\":
\"Player.PlayPause\", \"params\": { \"playerid\": 1 }, \"id\": 1}");
    strcpy(all_commands[8].command, "PAUSE");
    strcpy(all_commands[8].json_command, "{\"jsonrpc\": \"2.0\", \"method\":
\"Player.PlayPause\", \"params\": { \"playerid\": 1 }, \"id\": 1}");
    strcpy(all_commands[9].command, "STOP");
    strcpy(all_commands[9].json_command, "{\"jsonrpc\": \"2.0\", \"method\":
\"Player.Stop\", \"params\": { \"playerid\": 1 }, \"id\": 1}");
    strcpy(all_commands[10].command, "NEXT");
    strcpy(all_commands[10].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.GoTo\",\"id\":1,\"params\":{\"playerid\":1,\
"to\":\"next\"}}");
    strcpy(all_commands[11].command, "PREVIOUS");
    strcpy(all_commands[11].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.GoTo\",\"id\":1,\"params\":{\"playerid\":1,\
"to\":\"previous\"}}");
    strcpy(all_commands[12].command, "INCREMENT");
    strcpy(all_commands[12].json_command, "{ \"jsonrpc\": \"2.0\", \"method\":
\"Application.SetVolume\", \"params\": { \"volume\": \"increment\" }, \"id\": 1 }");
    strcpy(all_commands[13].command, "DECREMENT");
    strcpy(all_commands[13].json_command, "{ \"jsonrpc\": \"2.0\", \"method\":
\"Application.SetVolume\", \"params\": { \"volume\": \"decrement\" }, \"id\": 1 }");
    strcpy(all_commands[14].command, "MUTE");
    strcpy(all_commands[14].json_command, "{\"jsonrpc\": \"2.0\", \"method\":
\"Application.SetVolume\", \"params\": {\"volume\":0}, \"id\": 1}");
    strcpy(all_commands[15].command, "MAXIMUM");
    strcpy(all_commands[15].json_command, "{\"jsonrpc\": \"2.0\", \"method\":
\"Application.SetVolume\", \"params\": {\"volume\":100}, \"id\": 1}");


    FILE *fp;

    fp = popen(call_ultron, "r");
```

```
if(fp==NULL)
{
    perror("Error!");
    exit(1);
}


while(TRUE)
{
    ultron_called = FALSE;
    command_found = FALSE;

    //Verifica se o sistema foi chamado
    while (TRUE)
    {
        fscanf(fp, "%s", buffer);
        StrUp(buffer);
        if(!(strcmp(buffer, "ULTRON")))
        {

            system("aplay hello.wav");
            ultron_called = TRUE;
            break;
        }
    }


    //Caso o sistema tenha sido chamado, verifica o comando
    if(ultron_called)
    {
        fscanf(fp, "%s", buffer);
        StrUp(buffer);

        if(!(strcmp(buffer, "SHUTDOWN")))
        {
            pclose(fp);
            exit(0);
        }

        for(i=0; i < 16; i++)
        {
            if(!(strcmp(buffer, all_commands[i].command)))
            {
                strcpy(command, all_commands[i].json_command);
                command_found = TRUE;
```

```
                    break;
                }
            }

            if(command_found)
            {
                sprintf(curl_command, "curl -X POST -H \"Content-Type:
application/json\" -d '%s' localhost:8080/jsonrpc", command);
                system(curl_command);
            }

            else{
                system("aplay no_command.wav");
            }
        }
    }

    pclose(fp);
    return 0;
}



void StrUp(char *str)
{
    while(*str)
    {
        *str = toupper(*str);
        str++;
    }
}
```

10.3. Implementação do sistema Ultron em Python

```
import requests
import speech_recognition as sr

URL_KODI = "http://localhost:8080/jsonrpc?request={\"jsonrpc\": \"2.0\", \"method\":
"


r = sr.Recognizer()

while True:
```

```
    with sr.Microphone() as source:
        print("Waiting...")
        audio = r.listen(source)

    try:
        command = r.recognize(audio)

        if "Hey Ultron" in command.lower():
            print("Hi, my name is Ultron. How can I help you?")

            with sr.Microphone() as source:
                audio = r.listen(source)

            try:
                command = r.recognize(audio)

                if "Play" in command.lower():
                    COMMAND_KODI = "\"Player.PlayPause\", \"params\": { \"playerid\":
0 }, \"id\": 1}"

                elif "Pause" in command.lower():
                    COMMAND_KODI = "\"Player.PlayPause\", \"params\": { \"playerid\":
0 }, \"id\": 1}"

                elif "Stop" in command.lower():
                    COMMAND_KODI = "\"Player.Stop\", \"params\": { \"playerid\": 1 },
\"id\": 1}"

                elif "Initial Screen" in command.lower():
                    COMMAND_KODI = "\"Input.Home\"}"

                elif "Up" in command.lower():
                    COMMAND_KODI = "\"Input.Up\"}"

                elif "Down" in command.lower():
                    COMMAND_KODI = "\"Input.Down\"}"

                elif "Left" in command.lower():
                    COMMAND_KODI = "\"Input.Left\"}"

                elif "Right" in command.lower():
                    COMMAND_KODI = "\"Input.Right\"}"

                elif "Select" in command.lower():
                    COMMAND_KODI = "\"Input.Select\"}"
```

```
                elif "Back" in command.lower():
                    COMMAND_KODI = "\"Input.Back\"}"

                elif "Next" in command.lower():
                    COMMAND_KODI =
"\"Player.GoTo\",\"id\":1,\"params\":{\"playerid\":1,\"to\":\"next\"}}"

                elif "Previous" in command.lower():
                    COMMAND_KODI =
"\"Player.GoTo\",\"id\":1,\"params\":{\"playerid\":1,\"to\":\"previous\"}}"

                elif "Shutdown" in command.lower():
                    break

            except LookupError:
                print("Command inexistent")


            req = requests.get(URL_KODI + COMMAND_KODI)

    except LookupError:
        print("Sorry, you need to say 'Hey Ultron' first. Try again")
```