PONTO DE CONTROLE 4 – SISTEMAS EMBARCADOS

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

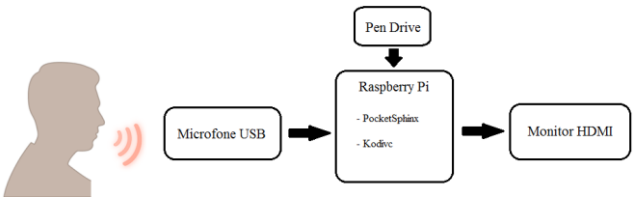

Figura 1: Diagrama geral do sistema

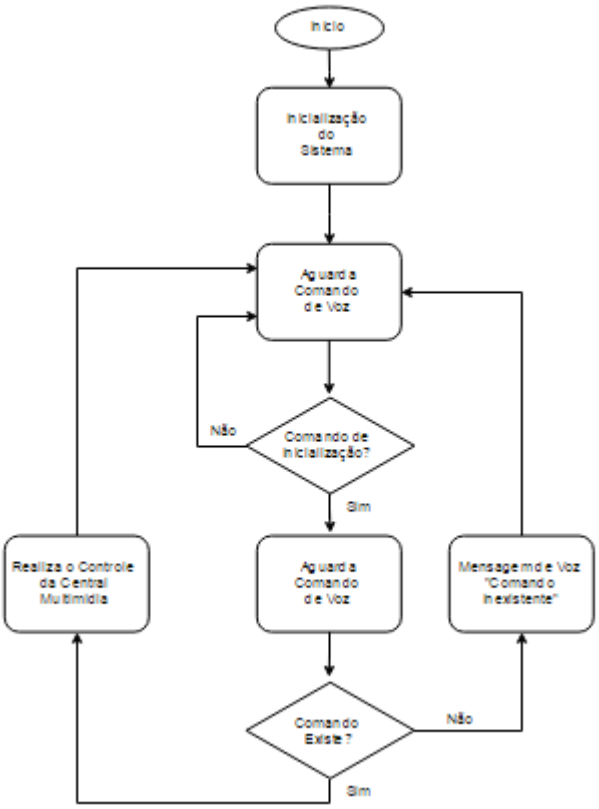

Figura 2: Fluxograma do sistema

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, é aberto um socket local entre o Ultron e o Kodi e os comandos são enviados pelo método de transporte POST, seguindo o seguinte formato [4]:

http://localhost:8080/jsonrpc?request=<url-encoded-request>

Onde o campo <url-encoded-request> é o comando a ser executado. Um exemplo de comando no formato adequado que pode ser utilizado é [5]:

{"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.

7. 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/?title=JSON-RPC_API#GET>

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

8. ANEXOS

8.1.Implementação do sistema em linguagem C

```
#include <stdio.h>
#include <stdbool.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>

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


//Inicializa os comandos do sistema
ultron_command * InitCommands();

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

//Determina qual comando será enviado para o Kodi
char *WhatCommand(char *voice_recognition, ultron_command *all_commands);


int main(){

	bool ultron_called, command_found;
	//char call_ultron[] = "pocketsphinx_continuous -adcdev plughw:1,0 -lm </path/to/1234.lm> -dict
</path/to/1234.dic> -inmic yes";
	char call_ultron[] = "python3 ultron.py";
	char buffer[500];
	char *command, *curl_command;

	ultron_command *all_commands;
	FILE *fp;

	all_commands = InitCommands();

	while(true){
		ultron_called = false;
		command_found = false;

		fp = popen(call_ultron, "r");
		if (fp==NULL){
			perror("Error!");
			exit(1);
```

```
		}

		//Verifica se o sistema foi chamado
		while (fscanf(fp, "%s", buffer)!=EOF){
			StrUp(buffer);

			if(!(strcmp(buffer, "NUMBER"))){
				system("aplay hello.wav");
				ultron_called = true;
				break;
			}
		}
		pclose(fp);

		//Caso o sistema tenha sido chamado, verifica o comando
		if(ultron_called){

			fp = popen(call_ultron, "r");
			if (fp==NULL){
				perror("Error!");
				exit(1);
			}

			while(fscanf(fp, "%s", buffer)!=EOF){
				StrUp(buffer);
				if(!(strcmp(buffer, "SHUTDOWN"))){
					exit(0);
				}

				command = WhatCommand(buffer, all_commands);

				if(strcmp(command, "NOT FOUND")){
					command_found = true;
					break;
				}
			}
			pclose(fp);

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

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

	free(all_commands);
	free(command);
	return 0;
}
```

```
ultron_command * InitCommands(){

	ultron_command *aux = calloc(16, sizeof(ultron_command *));

	strcpy(aux[0].command, "HOME");
	strcpy(aux[0].json_command, "{\"jsonrpc\":\"2.0\",\"method\":\"Input.Home\"}");
	strcpy(aux[1].command, "UP");
	strcpy(aux[1].json_command, "{\"jsonrpc\":\"2.0\",\"method\":\"Input.Up\"}");
	strcpy(aux[2].command, "DOWN");
	strcpy(aux[2].json_command, "{\"jsonrpc\":\"2.0\",\"method\":\"Input.Down\"}");
	strcpy(aux[3].command, "LEFT");
	strcpy(aux[3].json_command, "{\"jsonrpc\":\"2.0\",\"method\":\"Input.Left\"}");
	strcpy(aux[4].command, "RIGHT");
	strcpy(aux[4].json_command, "{\"jsonrpc\":\"2.0\",\"method\":\"Input.Right\"}");
	strcpy(aux[5].command, "SELECT");
	strcpy(aux[5].json_command, "{\"jsonrpc\":\"2.0\",\"method\":\"Input.Select\"}");
	strcpy(aux[6].command, "BACK");
	strcpy(aux[6].json_command, "{\"jsonrpc\":\"2.0\",\"method\":\"Input.Back\"}");
	strcpy(aux[7].command, "PLAY");
	strcpy(aux[7].json_command, "{\"jsonrpc\": \"2.0\", \"method\": \"Player.PlayPause\", \"params\": { \"playerid\": 1
}, \"id\": 1}");
	strcpy(aux[8].command, "PAUSE");
	strcpy(aux[8].json_command, "{\"jsonrpc\": \"2.0\", \"method\": \"Player.PlayPause\", \"params\": { \"playerid\": 1
}, \"id\": 1}");
	strcpy(aux[9].command, "STOP");
	strcpy(aux[9].json_command, "{\"jsonrpc\": \"2.0\", \"method\": \"Player.Stop\", \"params\": { \"playerid\": 1 },
\"id\": 1}");
	strcpy(aux[10].command, "NEXT");
	strcpy(aux[10].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.GoTo\",\"id\":1,\"params\":{\"playerid\":1,\"to\":\"next\"}}");
	strcpy(aux[11].command, "PREVIOUS");
	strcpy(aux[11].json_command,
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.GoTo\",\"id\":1,\"params\":{\"playerid\":1,\"to\":\"previous\"}}");
	strcpy(aux[12].command, "INCREMENT");
	strcpy(aux[12].json_command, "{ \"jsonrpc\": \"2.0\", \"method\": \"Application.SetVolume\", \"params\": {
\"volume\": \"increment\" }, \"id\": 1 }");
	strcpy(aux[13].command, "DECREMENT");
	strcpy(aux[13].json_command, "{ \"jsonrpc\": \"2.0\", \"method\": \"Application.SetVolume\", \"params\": {
\"volume\": \"decrement\" }, \"id\": 1 }");
	strcpy(aux[14].command, "MUTE");
	strcpy(aux[14].json_command, "{\"jsonrpc\": \"2.0\", \"method\": \"Application.SetVolume\", \"params\":
{\"volume\":0}, \"id\": 1}");
	strcpy(aux[15].command, "NUMBER");
	strcpy(aux[15].json_command, "{\"jsonrpc\": \"2.0\", \"method\": \"Application.SetVolume\", \"params\":
{\"volume\":100}, \"id\": 1}");

	return aux;
}


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

```
        }
}


char *WhatCommand(char *voice_recognition, ultron_command *all_commands){

        char *aux;
        int i;

        StrUp(voice_recognition);
        for(i=0; i < 16; i++){
                if(!(strcmp(voice_recognition, all_commands[i].command))){
                        aux = calloc(strlen(all_commands[i].json_command), sizeof(char));
                        strcpy(aux, all_commands[i].json_command);
                        return aux;
                }
        }

        aux = calloc(10, sizeof(char));
        strcpy(aux, "NOT FOUND");
        return aux;
}
```

8.2.Código executado em segundo plano

```
import speech_recognition as sr
r = sr.Recognizer()
with sr.Microphone() as source:
    audio = r.listen(source)
    print(r.recognize(audio))
```

8.3.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")
```