Projeto Final

Campainha Inteligente

Vitor Carvalho de Oliveira
Universidade de Brasília, Campus Gama
RA: 140165398
vitor.coliveiraa@gmail.com

Elpidio Cândido de Araújo Bisneto
Universidade de Brasília, Campus Gama
RA: 140137424
elpidio.araujo14@gmail.com

***Resumo* — O intuito de projeto consiste em desenvolver uma campainha inteligente capaz de fornecer recursos adicionais de segurança, comodidade e praticidade para o morador de uma residencia. O projeto utilizará a eletrônica embarcada do Raspberry Pi como plataforma de desenvolvimento buscando englobar a temática de Internet das Coisas (IoT)**

Palavras-chave—Internet das Coisas; Raspberry Pi; Sistemas embarcados, Campainha Inteligente.

I. INTRODUÇÃO

O avanço dos processos de fabricação tornou diversas tecnologias mais acessíveis e popularizadas, este fato, somado a ampla utilização da rede mundial de computadores tem fortalecido uma tendencia que está cada vez presente na vida de todos: o IoT (*Internet of Things*) ou Internet das Coisas.

Este termo refere-se a ideia de conectar objetos comuns do dia a dia à internet, viabilizando assim, o acesso e o controle remoto destes objetos, além de permitir que estes sejam provedores de serviços e recursos facilitadores para os usuários.

De acordo com a pesquisa realizada pela IDC, os projetos na área de IoT vão movimentar cerca de oito bilhões de dólares no Brasil em 2018, um crescimento de mais de 14% em comparação com 2017.

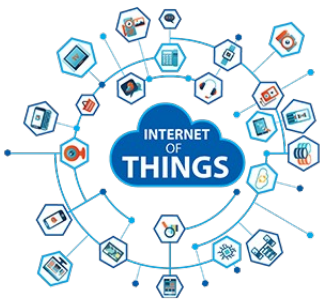

Figura 1: Figura ilustrativa envolvendo a temática de internet das coisas.

Atualmente, grandes empresas como Google, Microsoft, Sony, Ford, Apple entre outras têm lançado produtos que englobam o conceito de Internet das Coisas. Produtos estes que vão desde relógios sincronizados com celular, óculos de realidade aumentada, televisões com acesso a internet, até carros inteligentes.

Dentro do contexto domiciliar, equipamentos que permitem essa integração com a internet oferecem processos muitos mais automatizados, praticidade e comodidade para os moradores, além de poderem auxiliar na segurança e monitoramento das residencias. Em casas inteligentes que usam esse tipo de tecnologia é possível através do celular ou de um controle remoto controlar aparelhos domésticos, ascender ou apagar as luzes das lâmpadas e outras diversas funcionalidades. A ideia de Automação Residencial (AR) aliada ao conceito de internet das coisas ganha popularidade a cada dia diante dos inúmeros benefícios trazidos.

Seguindo esta tendencia, este projeto visa incluir a eletrônica embarcada em sistemas de campainhas residencias para oferecer comodidade à população.

Exitem alguns modelos de sistema de campainha inteligente no mercado. O “Ring” é um desses sistemas, consiste em uma campainha com câmera e sem fio fixada na porta da residência, ele transmite os dados para a nuvem e através do smartphone o morador pode acessá-la. É possível responder a quem está apertando a campainha e também por meio da câmera ver em tempo real quem está na porta.

Figura 2: Ring: Produto no mercado semelhante ao proposto pelo projeto.

Outro sistema é o Belle, um sistema mais sofisticado que utiliza Inteligência Artificial (IA) na campainha, ela vem equipada com uma câmera HD, áudio bidirecional, Wi-Fi e outros recursos. O sistema também faz reconhecimento facial e por meio de aplicativo no smartphone a IA direciona a chamada ao morador que o visitante está esperando.

Figura 3: Belle: Produto no mercado semelhante ao proposto pelo projeto.

II. JUSTIFICATIVA

O mercado da automação residencial mostra-se promissor no Brasil e no mundo. Segundo dados das Associação Brasileira de Automação Residencial, esse segmento prevê crescimento de 11,35% entre 2014 até 2020 em todo mundo e, no Brasil, há 300 mil lares que contam com essa tecnologia atualmente.

Essa demanda é reflexo de um modo de vida moderno que busca praticidade no dia a dia cada vez mais atarefado onde quase não se fica em casa. Além disso, dados mais recentes do IBGE mostram que há mais de 10,4 milhões de pessoas que moram sozinhas no Brasil, o que representa quase 15% dos domicílios do país. Diante disso, alternativas tecnológicas que permitam um monitoramento de casas à distância se tornam mais requisitadas.

Dentro deste contexto a campainha inteligente proposta neste projeto busca atender esse publico fornecendo soluções de monitoramento de visitantes em suas residências a distancia.

III. OBJETIVOS

A proposta deste projeto é modernizar e incluir a eletrônica embarcada utilizando o hardware Raspberry Pi em uma campainha residencial. O objetivo principal é permitir que o usuário do produto tenha mais recursos de segurança e de monitoramento do que o convencional. Ele poderá, por exemplo, saber da chegada de visitantes ou de encomendas mesmo que não haja ninguém em casa.

Com isso, a campainha inteligente visa ser um projeto de Internet das Coisas onde o sistema estará conectado à rede para fornecer ao morador a possibilidade de interagir mesmo longe, seja em uma conversa com o visitante por meio de uma ligação VoiP ou em uma mensagens de voz.

Figura 4: Representação ilustrativa do funcionamento da campainha inteligente.

IV. REQUISITOS

A campainha terá o funcionamento tradicional de uma campainha moderna em que o sistema irá informar ao morador, por meio de um sinal sonoro, quando o interruptor for acionado por algum visitante, além de mostrar, por meio de câmera, quem a tocou. Haverá ainda dois modos de operação: o modo I, em que após cessar a sinalização sonora, o sistema entenderá que não há ninguém em casa e irá gravar uma mensagem de áudio do visitante. Já no modo II, após cessar a sinalização sonora, o sistema irá iniciar uma chamada via VoIP para o celular do morador cadastrado no sistema.

Além disso, sempre que a campainha for acionada, uma foto ser capturada pela câmera do Raspberry . O objetivo é que tanto as fotos como as gravações de áudio sejam sincronizadas em um serviço de armazenamento em nuvem, de tal forma que o morador possa acessá-las pelo seu celular de qualquer lugar. Para fornecer tais recursos, o sistema precisará da conexão wifi do morador.

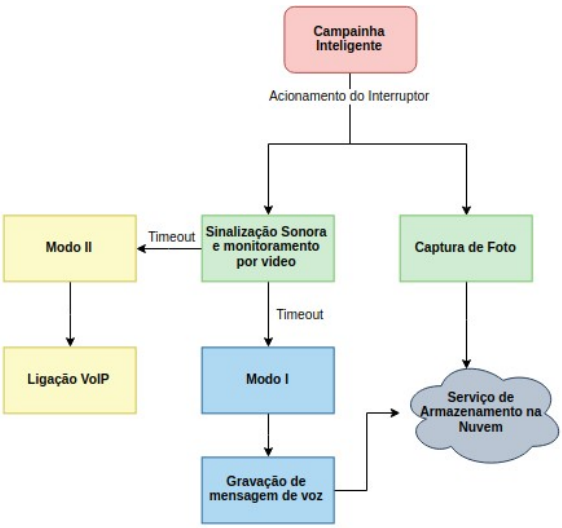

Figura 5: Diagrama de blocos simplificado da campainha inteligente.

Além destes recursos definidos, outros são avaliados no quesito viabilidade para aprimorar e ampliar os recursos da campainha inteligente. Por exemplo, incluir a tecnologia de detecção de face para somente iniciar uma gravação de voz ou uma chamada via VoIP caso seja identificada a presença de um rosto na câmera ou para abrir a porta caso o rosto do morador cadastrado seja reconhecido pelo sistema.

V. MATERIAIS UTILIZADOS

- Raspberry Pi 3 Modelo B;
- Raspberry Pi Câmera Versão 1.3;
- Placa de som USB com entrada pra microfone e fone de ouvido;
- Microfone de lapela P2;
- Botão de acionamento (*Push Button*) com dois terminais
- Jumpers Macho e Fêmea;
- Cartão Micro SD de 32Gb;
- Módulo GSM SIM800L;
- Bateria de celular – 3,85V/2000mAh;
- Alto-falante – 0,5W/8 Ohm;
- Cartão SIM to tipo Micro;
- Microfone de eletreto.

VI. DESCRIÇÃO DE HARDWARE

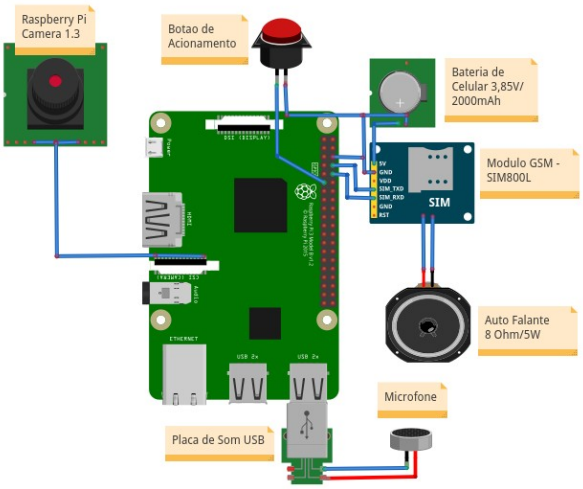

Figura 6: Esquemático ilustrativo das ligações dos componentes eletrônicos no Raspberry Pi.

Alimentou-se a placa Raspberry Pi com uma fonte com especificação de saída de 5V DC e 2,2 A, conectou-se a câmera na entrada CSI (*Camera Serial Interface*) própria da Raspberry, a placa de som em uma das entradas USB e o microfone com terminal P2 na entrada da placa de som destinada para microfones. Utilizou-se um fio para conectar a GPIO 17 (pino 11) em um dos terminais do botão de acionamento e outro para conectar ao terra (pino 6) no outro terminal do botão.

Para o módulo GSM, utilizou-se uma bateria de celular, visto que o módulo exige uma alimentação típica de 3,7 – 4,3V com capacidade de 1A. Conectou-se o pino RX do módulo com o pino TX da Raspberry (pino 8) e o TX do módulo com o RX da Raspberry (pino 10), por fim, conectou-se os terminais do alto-falante nos terminais destinados para saída de som do modulo GSM e os terminais do microfone do módulo nos terminais do microfone de eletreto.

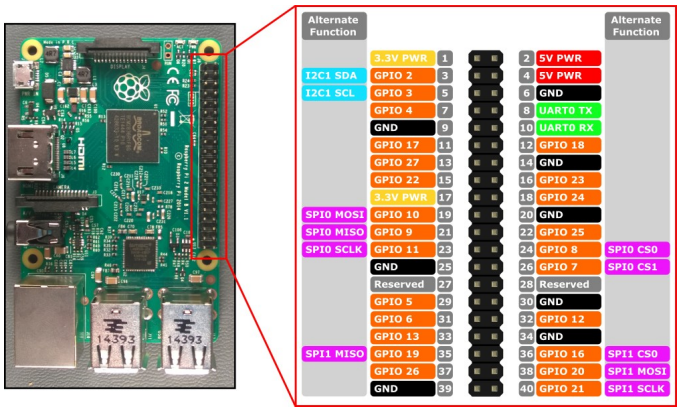

Figura 7: Mapeamento dos pinos da Raspberry Pi 3.

VII. DESCRIÇÃO DE SOFTWARE

Instalou-se na Raspberry Pi o sistema operacional Raspbian, que é uma distribuição baseada no sistema operacional Debian, versão *Stretch* com interface gráfica. Para isso, utilizou-se do instalador NOOBS versão 2.9.0, que facilita a instalação do sistema operacional.

Após os testes com os códigos separados, o código final, feito em C++, ficou definido conforme o diagrama de blocos abaixo:

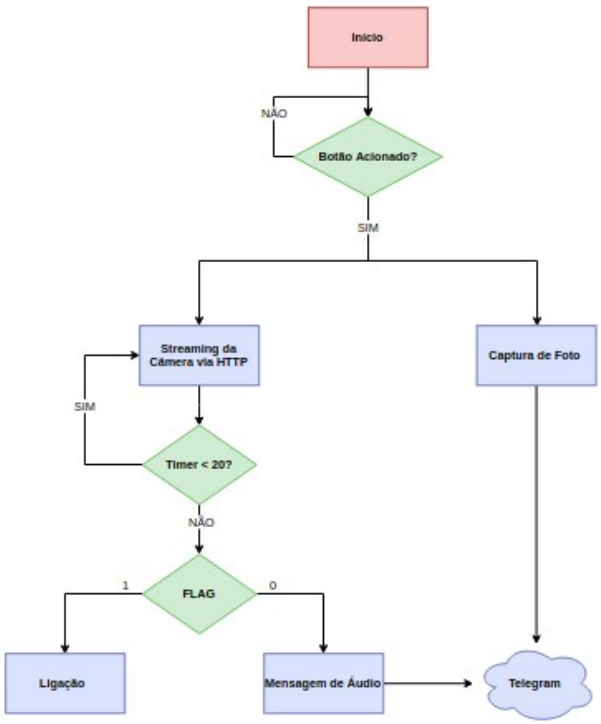

Figura 8: Diagrama de blocos ilustrativo do código do projeto.

O sistema inicializa definindo as bibliotecas necessárias, os pinos GPIOs que serão utilizados e um sinal de controle acionado por tempo, depois aguarda o acionamento do botão. Quando o botão é acionado, uma foto é capturada pela câmera, neste momento, o tempo do alarme é iniciado (20 segundos) e duas threads são criadas, a primeira envia a foto capturada e uma notificação (informando que houve um acionamento) para o chat do Telegram por meio de um bot criado para este projeto e a segunda que disponibiliza os quadros capturados pela câmera via *streaming* durante o tempo do alarme. Ambas as threads funcionam em paralelismo.

Quando o tempo do alarme é atingido, uma função de controle de alarme é ativada e o sistema pode, dependendo do valor de uma variável de controle, iniciar uma gravação de mensagem de voz de 10 segundos e a enviar para o telegram ou iniciar uma ligação por meio do módulo GSM utilizado.

A definição da variável de controle é feita passando um comando no chat do Telegram, no qual o mesmo é interpretado pelo bot e o valor da variável é definida. O monitoramento dos comandos enviados no chat do Telegram é feito por uma terceira thread, iniciada no começo do código, que fica ativa em um loop infinito.

Os detalhes do código relacionados as principais funcionalidades do sistema estão descritos abaixo:

a) Acionamento da Campainha:

Essa parte do código foi descrito utilizando a biblioteca WiringPi. Essa biblioteca facilita a configuração dos pinos GPIOs, para isso, fornece funções como: “pinMode” para configurar determinado pino como saída ou entrada; “pullUpDnControl” para configurar resistor de *pull up* ou *pull down* no pino desejado e “digitalRead” para realizar a leitura de nível lógico no pino. Para este código, as funções foram programadas da seguinte forma:

pinMode(0, INPUT) → Configuração do GPIO escolhido como entrada

pullUpDnControl(0, PUD_UP) → Definição do resistor de pull up na GPIO escolhida;

if(digitalRead(0) == LOW); → Verifica se o botão foi acionado

b) Gravação da mensagem de voz:

Para esta parte utilizou-se das ferramentas ALSA (*Advanced Linux Sound Architecture*) fornecidas pelo Raspbian para gravar uma mensagem de voz capturada pelo microfone conectado na placa de som USB.

Primeiro, executa-se o comando “arecord l” para verificar se a placa de som USB foi reconhecida pelo sistema operacional e a numeração do dispositivo. Depois, executa-se o comando :

- arecord -D plughw:X --duration=Y -f cd -vv ~/Z.wav

Sendo, “X” o número do dispositivo, descoberto pelo comando “arecord l” , “Y” a duração, em segundos, da gravação e “Z” o nome do arquivo de audio.

c) Enviar arquivo pro celular via Telegram:

Para esse requisito, utilizou-se o recurso fornecido pelo Telegram Bot API que consiste em uma interface baseada em HTTP que permite a criação de bots para o Telegram. Quando um bot é criado uma chave única de autenticação associada ao bot, chamada de *token*, é disponibilizada para o desenvolvedor.

Por meio dessa chave é possível realizar requisições do tipo GET e POST utilizando o comando “curl” permitindo assim, o envio de arquivos, por meio do bot, para um chat do Telegram. Os comandos possuem a seguinte estrutura básica:

curl -s -X POST https://api.telegram.org/<TOKEN>/

<COMANDO> - F chat_id<ID>

Em <TOKEN> deve ser inserido a chave de autenticação gerada ao criar o bot, em <COMANDO> deve ser inserido o comando referente a requisição desejada, o sítio da Bibliografia [9] próprio do Telegram, disponibiliza uma lista com todas os comandos possíveis. Em <ID> deve ser inserido ID do chat no qual o bot está inserido. Para descobrir esse ID basta acessar o sítio: https://api.telegram.org/bot<TOKEN>/getUpdates.

Para este projeto foram utilizados os seguintes comandos:

- sendAudio;
- sendMessage;
- sendPhoto.

d) Controle da Raspberry por meio do Telegram:

Explorando ainda mais os recursos do API do Telegram, criou-se uma *Thread* que é executada em paralelo ao restante das funcionalidades do projeto, que monitora, por intermédio do Bot, os comandos enviados no chat do Telegram.

Para este projeto foram criados 6 comandos:

- /capture: Captura um foto pela câmera da Raspberry e envia no chat do Telegram;
- /stream_on: Ativa o stream da câmera;
- */stream*_off: Desabilita o stream da câmera;
- /rec_flag: Habilita a flag para gravar uma mensagem de voz após o acionamento da campainha;
- /call_flag: Habilita a flag para realizar uma ligação para o número cadastrado no sistema após o acionamento da campainha;
- /call: Realiza uma ligação imediata pro número cadastrado.

Essa parte do código utiliza a biblioteca “tgbot/tgbot.h”, dispobilizada no github da Bibliografia [8].

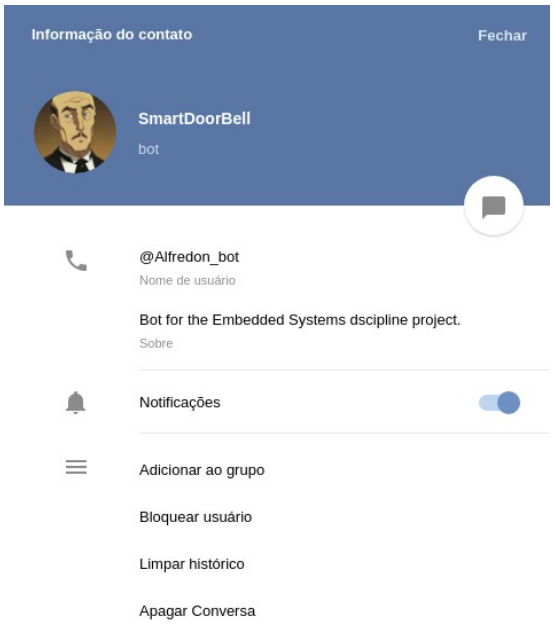

Figura 9: Bot criado para o projeto.

e) Streaming da câmera:

Para essa funcionalidade, é criada uma "thread" que funciona no modelo cliente-servidor no qual o fluxo de informação ocorre via "socket" e protocolo HTTP/TCP. Neste código, o servidor local disponibiliza os quadros processados em uma página Web que pode ser acessada pelo cliente por meio do IP do servidor local e a porta pre definida.

A codificação dos quadros capturados pela câmera é feita no formato MJPEG (*Motion JPEG*), ou seja, cada quadro antes de ser transmitido é comprimido no formato JPEG. Essa técnica, por não usar usar compressão inter-quadro, fornece transmissões com baixo período de latência e a qualidade da imagem depende da qualidade do codificador e da disponibilidade de largura de banda para transmissão.

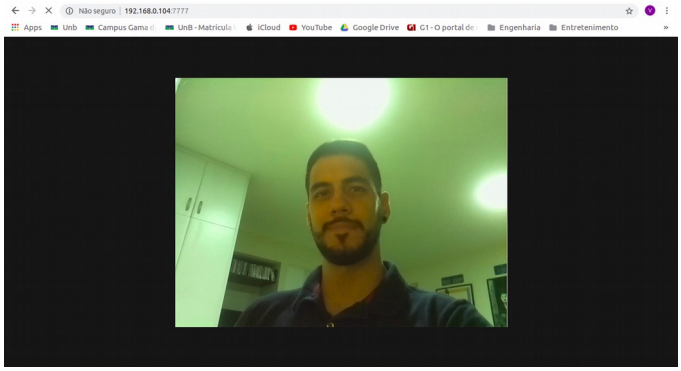

Figura 10: Resultado obtido com o streaming .

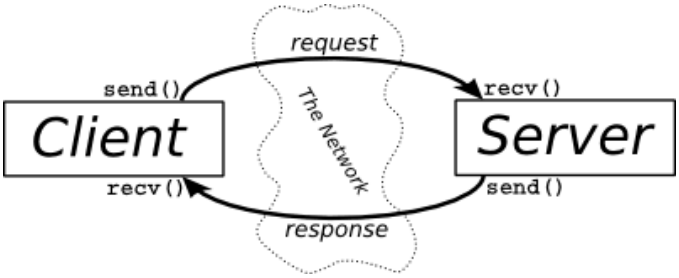

Figura 11: Fluxograma ilustrativo do código cliente servidor.

f) Ligação via módulo GSM:

Para essa parte do código, utilizou-se comandos do tipo AT também conhecidos como Hayes, que consistem em um linguagem desenvolvida em 1981 por Dennis Hayes para estabelecer comunicação com modems. Os comandos são passados da Raspberry para o módulo GSM via comunicação UART. Para realizar a ligação para o celular de um dos moradores, utilizou-se o seguinte comando:

- ATDXXXXXXXXX;

Sendo XXXXXXXXX o número do morador. Esse comando pode retornar os seguintes atributos:

- NO DIALTONE: Significa que não foi possível estabelecer o tom de discagem, condição necessária para iniciar uma chamada;
- BUSY: Quando o celular alvo estiver ocupado;
- NO CARRIER: Quando não há resposta da estação remota;
- OK: Quando houver comunicação, indica que a ligação iniciará.

VIII. Estrutura do projetos

A estrutura foi projetada para ser impressa em 3D, onde comportará a câmera, a Raspberry Pi, os botões, além da placa de áudio e microfone. O sistema foi pensado para ser compacto e com o menor tamanho possível.

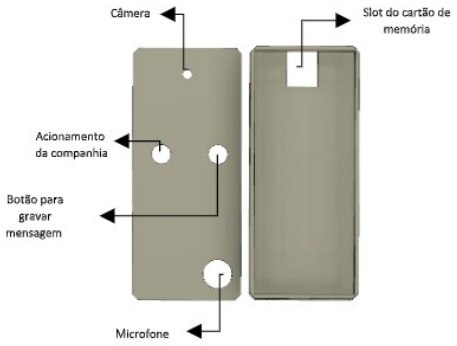

Figura 12: Estrutura projetada para o projeto – Visão frontal.

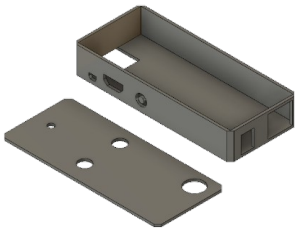

Figura 13: Estrutura projetada para o projeto – Visão em perspectiva.

A estrutura final foi feita em madeirite com o intuito de baratear o custo e ser mais acessível. A estrutura da caixa tem tamanho de 21,5x15,5x5,5cm. A Figura abaixo mostra como ficou a estrutura.

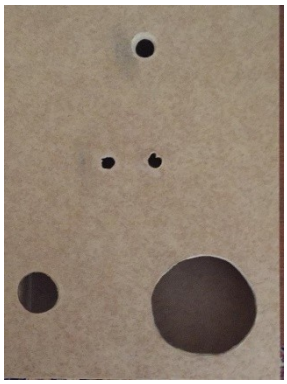

Figura 14: Estrutura do projeto feita em madeirite

IX. CUSTO DO PROJETO

Tabela 1: Custo do Projeto.

Material	Preço (R$)
1 Raspberry Pi 3 Modelo B	193,90
1 Raspberry Pi Câmera Versão 1.3	50,00
1 Placa de som USB com entrada pra microfone e fone de ouvido	16,50
1 Microfone de lapela P2	9,89
2 Botão de acionamento (*Push Button*) com dois terminais	0,40
40 Jumpers Macho e Fêmea	12,90
1 Cartão Micro SD de 32Gb	30,00
1 Módulo GSM SIM800L	27,58
1 Alto-falante – 0,5W/8 Ohm	9,90
1 Caixa de madeira 21,5x15,5cm	5,00
Total	R$ 356,07

X. CONSIDERAÇÕES FINAIS

Após algumas pesquisas sobre ligação via VoIP utilizando o Raspberry, notou-se que seria necessário criar um servidor dedicado para esta funnção o que iria comprometer as outras funcionalidades do sistema, como alternatva, decidiu-se utilizar o módulo GSM para realizar ligações. Apesar de ter obtido sucesso em realizar ligação, observou-se instabilidade referente ao sinal do módulo. Em locais com baixa cobertura de sinal de telefonia era dificil conseguir estabelecer uma ligação, além disso, não foi possível transmitir a voz captada pelo microfone de eletreto, acredita-se que seria necessário um circuito amplificador antes de enviar o sinal pro módulo.

A requisito inicial de enviar os arquivos para um servidor na núvem, como por exemplo o DropBox ou Google Drive, foi subtituido pelo envio direto ao aplicativo do Telegram que possibilitou maior facilidade no acesso aos arquivos por parte do usuário e recursos adicionais, como por exemplo, o controle da Raspberry por meio de comandos criados e interpretados pelo Bot do Telegram.

Em relação ao recurso de detecção de rosto, apesar de ter obtido exito, conforme observa-se na figura 14 dos Anexos, notou-se um elevado percentual de processamento da Raspberry (em torno de 85 a 95%) além de uma pequena lentidão nos quadros processados e transmitidos pelo streaming, que se intensificava conforme limitações na qualidade da rede, além disso, em certos momentos, o código do streaming entrava em conflito com o de detecção de rosto, visto que, ambos compartilhavam os mesmos quadros capturados pela câmera. Diante desses resultados, obtou-se por retirar essa funcionalidade do projeto.

O custo final do projeto se comparado com produtos semelhantes se mostrou acessivel, visto que ele também se torna mais barato do que implementar um smartphone como campanhia, por exemplo.

Como melhorias para o sistema, sugere-se:

1. Implementar notificações sonoras/visuais, para o possível visitante, indicando o momento de gravar a menssagem de voz e o momento em que inicia-se a ligação pro morador da residencia;
2. Confeccionar a estrutura do modelo 3D do projeto;
3. Ampliar a lista de comandos, fornecendo novos recursos pro Bot;
4. Integrar o sistema com o mecanismo de abertura da fechadura.

XI. BIBLIOGRAFIA

[1] “Sonho da “casa inteligente” está mais próximo da classe média”, Revista Exame, Agosto de 2018

[2] “48% dos brasileiros que moram sozinhos tomaram essa decisão por vontade própria”, Site do Programa Nacional de Desenvolvimento do Varejo 2016-2018 da Confederação Nacional dos Dirigentes Lojistas e do Sebrae.

[3] “Internet das Coisas’: entenda o conceito e o que muda com a tecnologia”, Portal TechTudo , 2014.

4] Roberto Hammerschmidt. Ring, a campainha inteligente com vídeo que se conecta com o seu smartphone, 2018. Em <https://www.tecmundo.com.br/produtos/86696-ring-campainha-inteligente-video-conecta-com-smartphone.htm. Ultimo acesso: 19/10/2018.>

[5] Raquel Freire, Campainha inteligente Belle tem câmera e 'conversa' com suas visitas, 2018. Em <https://www.techtudo.com.br/noticias/2018/02/campainha-inteligente-belle-tem-camera-e-conversa-com-suas-visitas.ghtml. Ultimo acesso: 19/10/2018>

[6] “Internet das coisas movimentará US$ 8 bi no Brasil em 2018, estima IDC”, Portal Valor, Janeiro de 2018.

[7] Comandos AT do módulo SIM800L/900, Disponível em: <http://www.rhydolabz.com/documents/gps_gsm/sim900_at_commands.pdf>

[8] Biblioteca pro Telegram API. Disponível em: <https://github.com/reo7sp/tgbot-cpp>

[9] Lista de recursos e comandos do tipo GET e POST para Bots do Telegram, Disponível em: <https://core.telegram.org/bots/api>

XII. ANEXOS

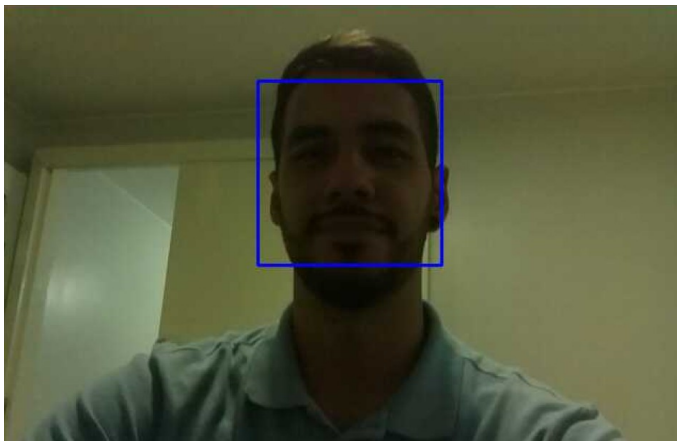

Figura 15: Funcionamento do código de detecção de rosto.

Código Principal:

```
#include <stdio.h>
#include <wiringPi.h>
#include <pthread.h>
#include <time.h>
#include <signal.h>
#include <stdlib.h>
#include "MJPEGWriter.h"
#include <tgbot/tgbot.h>

pthread_t streamer, notification, telegram_tbot, tpisca_led;
using namespace std;
int control,flag=0;
MJPEGWriter test(8080);

void* pisca_led(void*){

	while(1)
		{
			digitalWrite(1, HIGH); // liga o pino 1
			delay(1000);		// espera 1 segundo
			digitalWrite(1, LOW);  // desliga o pino 1
			delay(1000);			// espera 1 segundo
		}
	return NULL;

}

void* push_notification(void*){

	system("curl -s -X POST https://api.telegram.org/bot785342648:AAEJRJ9gwGZvgA86lXmd83U11IgXYOzgugo/sendMessage -d chat_id=-312166514 -d text='SmartDoorBell - Acionamento da Campainha'");
```

```
        delay(20);
        system("curl -s -X POST 'https://api.telegram.org/bot785342648:AAEJRJ9gwGZvgA86lXmd83U11IgXYOzgugo/sendPhoto' -F chat_id=-312166514 -F photo='@/home/pi/MJPEGWriter/FotoSmartDoorbell.jpg'");
        return NULL;
}

void* streaming(void*)
{
printf("Streaming Disponivel\n");

   VideoCapture cap;
   bool ok = cap.open(0);
   if (!ok)
   {
      printf("no cam found ;(.\n");
      pthread_exit(NULL);
   }

   Mat frame;
   cap >> frame;
   test.write(frame);
   frame.release();
   test.start();

   while(cap.isOpened() && (control == 0))
   {
        cap >> frame;
        test.write(frame);
        frame.release();
   }

   printf("Fim do Loop");
   cap.release();

   //exit(0);
   return NULL;
}

void* telegram_bot(void*){

        TgBot::Bot bot("785342648:AAEJRJ9gwGZvgA86lXmd83U11IgXYOzgugo");

        bot.getEvents().onCommand("capture", [&bot](TgBot::Message::Ptr message) {
        bot.getApi().sendMessage(message->chat->id, "Uma foto está sendo capturada e chegará em breve.");
                                system("raspistill -o /home/pi/MJPEGWriter/FotoSmartDoorbell.jpg");
        system("curl -s -X POST 'https://api.telegram.org/bot785342648:AAEJRJ9gwGZvgA86lXmd83U11IgXYOzgugo/sendPhoto' -F chat_id=-312166514 -F photo='@/home/pi/MJPEGWriter/FotoSmartDoorbell.jpg'");
   });

                bot.getEvents().onCommand("call", [&bot](TgBot::Message::Ptr message) {
      bot.getApi().sendMessage(message->chat->id, "Ligação iniciada.");
   system("python sim800_call.py");
   });

        bot.getEvents().onCommand("call_flag", [&bot](TgBot::Message::Ptr message) {
   flag = 1;
        bot.getApi().sendMessage(message->chat->id, "Modo Mensagem de Voz Desligado");
        bot.getApi().sendMessage(message->chat->id, "Modo Ligação Ligado");
   });

        bot.getEvents().onCommand("rec_flag", [&bot](TgBot::Message::Ptr message) {
   flag = 0;
        bot.getApi().sendMessage(message->chat->id, "Modo Ligação Desligado");
        bot.getApi().sendMessage(message->chat->id, "Modo Mensagem de Voz Ligado");
   });

        bot.getEvents().onCommand("stream_on", [&bot](TgBot::Message::Ptr message) {
   control = 0;
      bot.getApi().sendMessage(message->chat->id, "Stream Disponivel");
   pthread_create(&streamer, NULL, streaming, NULL);
   });

        bot.getEvents().onCommand("stream_off", [&bot](TgBot::Message::Ptr message) {
   control = 1;
      bot.getApi().sendMessage(message->chat->id, "Stream Indisponivel");
   });

   try {
        printf("Bot username: %s\n", bot.getApi().getMe()->username.c_str());
      TgBot::TgLongPoll longPoll(bot);
      while (true) {
         printf("Long poll started\n");
         longPoll.start();
      }
   } catch (TgBot::TgException& e) {
      printf("error: %s\n", e.what());
   }

return NULL;
```

```
}

void tratamento_alarme(int sig)
{
	pthread_cancel(tpisca_led);
	control = 1;
	//cap.release();
	printf("Streaming Finalizado\n");

	if(flag==0){
		digitalWrite(1, HIGH); // liga o pino 1
		printf("Gravando Mensagem de Voz\n");
		system("arecord -D plughw:1 --duration=10
-f cd -vv ~/MJPEGWriter/rectest.wav");
		delay(10500);
		digitalWrite(1, LOW);  // desliga o pino 1
		system("curl -s -X POST
https://api.telegram.org/bot785342648:AAEJRJ9gwGZvgA86
lXmd83U11IgXYOzgugo/sendAudio -F chat_id=-312166514
-F audio='@/home/pi/MJPEGWriter/rectest.wav'");
	}

	if(flag==1){
		system("python sim800_call.py");
	}

}

int main()
{
	pthread_create(&telegram_tbot, NULL,
telegram_bot, NULL);
	wiringPiSetup();	// inicia biblioteca wiringPi
	pinMode(0, INPUT);	// configura pino 7 como
entrada
	pinMode(1, OUTPUT);
	pullUpDnControl(0, PUD_UP); // configura resistor
pull-up no pino 7
	signal(SIGALRM, tratamento_alarme);

	while(1)
	{
		if(digitalRead(0) == LOW)	// detecta se
a chave foi pressionada
		{
			delay(20);
			control = 0;
			pthread_create(&tpisca_led,
NULL, pisca_led, NULL);
			system("raspistill -o
/home/pi/MJPEGWriter/FotoSmartDoorbell.jpg");
			pthread_create(&notification,
NULL, push_notification, NULL);
			//pthread_join(notification,NULL);
			pthread_create(&streamer, NULL,
streaming, NULL);
			alarm(20);
			while(digitalRead(0) == LOW); //
aguarda enquato chave ainda esta pressionada

			delay(20);
		}
	}
return 0;
}
```