Stelia

Magic Mirror

Stefany Santos Aquino
Universidade de Brasília
Faculdade do Gama - FGA
Brasília, Brasil
stepaquino@gmail.com

Júlia Pessoa Souza
Universidade de Brasília
Faculdade do Gama (FGA)
Brasília, Brasil
juliapessoasouza@gmail.com

I. Introdução

Autoestima é uma parte fundamental da felicidade. Ter confiança em si ajuda a tomar boas decisões, o que melhora o desenvolvimento pessoal e a forma de lidar com as pessoas, no mundo em que vivemos é cada vez mais difícil obter a aceitação tanto social como própria [1]. Atualmente a tecnologia nos cerca de inúmeras maneiras, inclusive com relação a saúde [6]. Internet das Coisas é o termo utilizado para descrever diversos objetos que se conectam a internet e se comunicam mutuamente [3]. Com isso, é possível desenvolver equipamentos úteis para a vida das pessoas tornando-as melhores de algum jeito [5].

II. Objetivo

O projeto tem a finalidade de criar um espelho inteligente conectado a Internet. No próprio reflexo, quando detectar a presença de uma pessoa, o espelho mostra data, hora, temperatura, frases, lembretes e compromissos. Com um microfone e uma câmera embutidos, o projeto também possui as funcionalidades de comando de voz e de tirar fotos. A internet é utilizada para buscar informações de tempo e clima na cidade, assim como mandar fotos por e-mail. As frases e lembretes poderão ser descritos na tela ou reproduzidos pelo alto falante [4].

III. Justificativa

A criação do projeto do espelho interativo foi motivada pelo o quanto a tecnologia pode influenciar a vida das pessoas, principalmente relacionado com sua autoestima. O espelho é um utensílio comumente usado no cotidiano das pessoas, com esse projeto ele deixará de ser um objeto comum para ser um objeto inovador adaptado à tecnologia inovadora de internet das coisas (IoT).

O uso da Raspberry Pi possibilita criar um sistema embarcado para a finalidade especificada, o que não seria possível usando um smartphone, por exemplo. Várias funções e aplicativos do mesmo ficariam inutilizáveis, tornado-se um prejuízo perante a tecnologia.

O diferencial seria enquanto você penteia seu cabelo, escova seus dentes pela manhã ou simplesmente quando você se observa no espelho, você possa ao mesmo tempo olhar as horas, saber as notícias do dia, ou simplesmente ouvir uma música.

Todas essas funcionalidades em um só objeto permite que o usuário economize seu precioso tempo. No mundo atual, um dos bens mais precioso é o tempo, e toda forma de poupá-lo é muito bem vinda.

IV. Benefícios

Erroneamente, podemos ser levados a pensar que o projeto do espelho interativo possui um leque restrito de benefícios. Porém, basta analisar o cotidiano de uma pessoa, enquanto se arruma em frente ao espelho para iniciar o seu dia, por exemplo, ela se depara com inúmeras funcionalidades no mesmo objeto. Ela pode se encorajar com mensagens motivacionais, analisar sua agenda diária. Também é possível pensar em um idoso, que pode ser avisado qual o horário de tomar seu remédio.

V. Requisitos

Os requisitos podem ser divididos entre Hardware e Software.

A. Requisitos de Hardware

- Raspberry Pi;
- Cartão de memória SD;

- Monitor de Led;
- Acrílico espelhado;
- Cabo HDMI;
- Microfone
- Câmera
- Conexão internet;
- Sensor de presença.

O projeto precisará da Raspberry como plataforma para rodar a programação do sistema dedicado à aplicação. Será necessário controlar com a Raspberry um monitor, que servirá como uma interface de usuário, e para isso a saída HDMI da placa será usada com um cabo de conexão ligado à tela. O monitor será espelhado, o que será responsável pela aparência de espelho do dispositivo inteligente a ser desenvolvido. Também será utilizado um microfone para ativação do comando de voz e um pequeno alto falante para as saídas de áudio relativas ao projeto. O sensor de presença ajudará na economia de bateria, detectando se há alguém próximo ao espelho.

B. Requisitos de Software

- Sistema Operacional para placa Raspberry
- Bibliotecas para comando de voz.
- Programa para execução de arquivos de áudio.
- Códigos para sensores.

O sistema operacional que será incluído na placa será o Raspbian, foi escolhido por atender a funcionalidade demandada pelo projeto. Esse sistema operacional será responsável por gerenciar e ordenar a execução do software.

C. Sensor de presença

O sensor de presença utilizado foi o PIR (Passive Infrared Sensor) HC-SR501. Esse sensor verifica se há movimento a partir da variação da radiação da luz infravermelha em um determinado ambiente. O sensor descrito atua com saídas digitais, em que quando acionado seu nível lógico ficará alto e caso contrário ele permanecerá em nível lógico baixo. O chip que compõe o sensor atua como amplificador e comparador de modo que ao comparar e detectar variação da luz infravermelha o sensor apresentará saída em nível lógico alto, por um tempo determinado pela regulação do trimpote descrito na Figura 1 como "*Delay Time Adjust*". Já o outro trimpote permite regular com relação a distância de acionamento de movimento, o qual pode ser observado na Figura 1 como "*Distance Adjust*", quanto mais no sentido horário, maior será a sensibilidade adquirida.

Figura 1 - Sensor PIR.

A ligação do circuito do sensor pode ser vista na Figura 4, em que foi realizada da seguinte forma: o pino 4 da Raspberry, que possui como saída a tensão de 5V, de cor vermelha, foi ligado ao pino de VCC do sensor. O pino 11(GPIO 17) , entrada/saída digital, foi concetado ao pino central do sensor o qual é o pino de saída. O pino 34 da Raspberry (GND) foi interligado ao pino de GND do sensor.

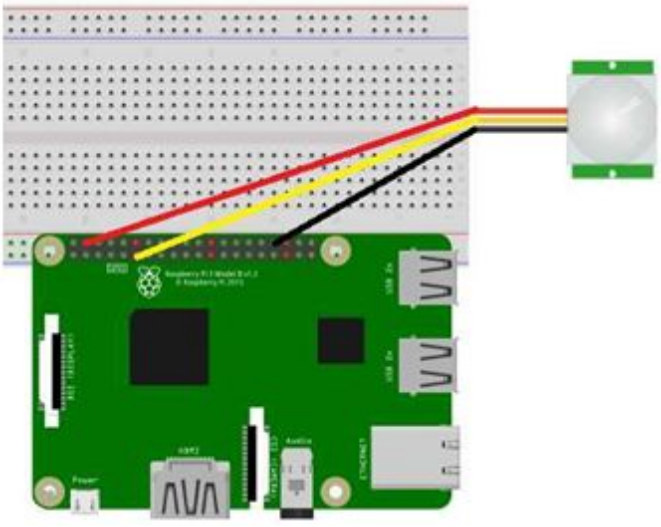

Figura 2 - Esquemático de ligações do sensor.

VI. DESENVOLVIMENTO

Para o desenvolvimento do protótipo do projeto foram utilizadas bibliotecas prontas no Raspberry Pi. Primeiramente foi instalado o Raspbian no microcontrolador, um Sistema Operacional livre inspirado no Debian, que facilita a interação e a programação do dispositivo. O Download do sistema foi feito no site oficial e, para a instalação na Raspberry Pi, utilizou-se um cartão de memória com 16gb.

Após o Sistema Operacional ser instalado, pelo terminal do microprocessador, foi possível atualizar a distribuição e o firmware e instalar a biblioteca MagicMirror[2]. Esta biblioteca

foi desenvolvida e disponibilizada pelo criador do primeiro MagicMirror para ser utilizada e aprimorada. É uma biblioteca feita para ser modular, pode-se modificar e adicionar módulos para uso customizado [7].

O módulo principal controla a interface gráfica mostrada na tela do espelho, já os outros são aplicativos que compõem a tela de espelho em si. Existem os módulos que são padrões no projeto inicial e os módulos acrescentados de acordo com a necessidade de cada pessoa que reproduz o projeto. No caso do espelho inteligente os módulos a serem acrescentados seriam as aplicações de comando de voz, sensoriamento e o script de reprodução de áudio. Um dos módulos padrões que serão mantidos para o projeto será o módulo de relógio e de calendário, por exemplo.

O sensor de presença PIR foi testado e adicionado ao projeto utilizando um código em linguagem C. Foram baixados arquivos de extensão .sh para ativar e desativar o monitor, os quais foram colocados na pasta root da Raspberry Pi. Esses arquivos contém o comando "vgencmd display_power" do Linux, que vai para 0 se o monitor for desligado e para 1 se for reiniciado. A programação chama o comando contido no arquivo para apagar o monitor caso o sensor não detecte presença por mais de 10 segundos e, ativar o monitor caso detecte alguma presença e o monitor estiver apagado. O código pode ser visualizado no ANEXO I.

O arquivo do código foi colocado na pasta autostart do RaspBerry Pi que executa assim que é inicializado. Portanto, assim que o microcontrolador é ligado ele executa automaticamente o MagicMirror e o código do sensor.

Foi feita a implementação de uma câmera com um microfone no Magic Mirror. A webcam de Playstation 3 já possui um microfone embutido e cabo USB. Para se utilizar estes módulos foi feito o gitclone dos códigos em JavaScript dos módulos camera e VoiceControl. O Microfone foi testado no terminal da Raspberry e, pelo site [9] foi possível gravar áudios e linkar com comandos, transformando-os em arquivos com extensão .pmdl. Foram obtidos os arquivos para comandos de mostrar câmera, desligar câmera e selfie. Os nomes utilizados nos arquivos e os comandos foram atualizados no código do módulo.

A câmera também foi testada pelo terminal da Raspberry. O código do módulo da câmera foi atualizado para enviar a foto para um e-mail. O e-mail e a senha também foram inseridos neste código.

A estrutura foi feita em madeira para a moldura em volta do monitor. Já o espelho foi feito utilizando-se uma placa de acrílico com um papel filme espelhado.

VII. Resultados

Com o Sistema Operacional em funcionamento e as bibliotecas instaladas o sistema foi testado e pode-se ver o resultado pela imagem.

Figura 3: Magic Mirror com estrutura, sensor de presença e câmera com microfone..

O sensor de presença PIR foi testado e funciona corretamente, quando não há movimento por mais de 10 segundo, o monitor apaga, já quando o sensor detecta movimento novamente o monitor é religado com o Magic Mirror.

Os testes da câmera e do microfone funcionaram corretamente na Raspberry. O áudio utilizado foi reconhecido, porém, não foi possível implementá-los no espelho.

Figura 4: Teste de imagem feita pela câmera com a Raspberry.

Apesar dos testes bem sucedidos, as notificações de comandos não apareciam no espelho, e os comandos de voz não funcionavam. Sem os comandos de voz não havia como utilizar a câmera junto ao espelho.

Referências

[1] M. Marcus, A Autoestima e sua influência no desenvolvimento profissional. <http://marcusmarques.com.br/comportamento/autoestima-influencia-desenvolvimento-profissional/> Acessado em 5 de Setembro de 2018.

[2] Raspberry Pi Smart Mirror: <https://hackaday.io/project/13466-raspberry-pi-smart-mirror> Acessado em 05 de Setembro de 2018.

[3] Associação Brasileira de Internet das Coisas, O que é internet das coisas?- 16 de Janeiro, 2017: <http://abinc.org.br/www/2017/01/16/o-que-e-a-internet-das-coisas/> Acessado em 5 de Setembro de 2018.

[4] B. Max, "My bathroom mirror is smarter than yours" <https://medium.com/@maxbraun/my-bathroom-mirror-is-smarter-than-yours-94b21c6671ba> Acessado em 5 de setembro de 2018.

[5] "O que é Internet das Coisas? (Internet of Things)" <https://www.infowester.com/iot.php> Acessado em 5 de setembro de 2018.

[6] P. Douglas "Internet das Coisas: o futuro da saúde já começou a ser monitorado." <https://canaltech.com.br/internet-das-coisas/internet-das-coisas-o-futuro-da-saude-ja-comecou-a-ser-monitorado-104926/> Acessado em 5 de setembro de 2018.

[7] <https://magicmirror.builders/> Acessado em 5 de setembro de 2018.

[8] <https://www.postscapes.com/diy-smart-mirrors/> Acessado em 5 de setembro de 2018.

[9] <https://snowboy.kitt.ai/> Acessado em 1 de dezembro de 2018.

ANEXO I

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/time.h>
#include <wiringPi.h>
#include <pthread.h>

#define LOOP_DELAY 1000
#define DEBUG   0
#define TIMESTAMP_LENGTH 27
#define PIR_PIN 17
#define SIGNAL  1
#define DISPLAY_OFF 0
#define DISPLAY_ON  1
#define DEFAULT_TIMER_SEC 180

const char *CMDS[] = {
                "/home/pi/bin/monitor-off.sh",
                "/home/pi/bin/monitor-on.sh"
        };

char timestamp[TIMESTAMP_LENGTH];
int timer = DEFAULT_TIMER_SEC;

void printUsage(void);
int init(void);
void exInt0_ISR(void);
void display_off(void);
void display_on(void);

void printUsage() {
        fprintf(stderr, "Usage: piscreenctrl\n"
            "Program listens on GPIO [PIR_PIN] for HIGH signal and executes turns on/off display.\n");
        exit(EXIT_FAILURE);
}

int init(void) {
        if (wiringPiSetup()<0) {
                fprintf(stderr, "Unable to setup wiringPi: %s\n", strerror(errno));
                exit(EXIT_FAILURE);
                return 1;
        }
        wiringPiISR(PIR_PIN, SIGNAL, &exInt0_ISR);
        if (DEBUG) {
                createTimestamp(timestamp);
                fprintf(stdout,"[%s] [piscreenctrl] init() pir=%d signal=HIGH)\n", timestamp, PIR_PIN);
                fflush(stdout);
        }
        return 0;
}

void *countdown(void *threadid) {
   long tid;
   tid = (long)threadid;
   if (DEBUG) {
                createTimestamp(timestamp);
                fprintf(stdout,"[%s] [piscreenctrl] countdown thread started threadid=%d\n", timestamp, tid);
        }
        while (1) {
                delay(1000); // 1000ms
                timer = timer - 1;
                timer = timer<0 ? 0 : timer;
        }
   pthread_exit(NULL);
}

 void display_off(void) {
                createTimestamp(timestamp);
                fprintf(stdout, "[%s] display_off()\n", timestamp);
                fflush(stdout);
                system(CMDS[DISPLAY_OFF]);
 }

 void display_on(void) {
                createTimestamp(timestamp);
                fprintf(stdout, "[%s] display_on()\n", timestamp);
                fflush(stdout);
                system(CMDS[DISPLAY_ON]);
 }
```

```
void exInt0_ISR(void) {
        if (DEBUG) {
                createTimestamp(timestamp);
                fprintf(stdout, "[%s] exInt0_ISR(%d): Motion detected. Reseting timer to %dsec\n", timestamp, PIR_PIN, DEFAULT_TIMER_SEC);
                fflush(stdout);
        }
        timer = DEFAULT_TIMER_SEC;
        return;
}

int createTimestamp(char* buffer) {
  int millisec;
  struct tm* tm_info;
  struct timeval tv;
  char buffer2[TIMESTAMP_LENGTH];

  gettimeofday(&tv, NULL);

  millisec = lrint(tv.tv_usec/1000.0);
  if (millisec>=1000) {
    millisec -=1000;
    tv.tv_sec++;
  }

  tm_info = localtime(&tv.tv_sec);
     strftime(buffer, TIMESTAMP_LENGTH, "%Y-%m-%d %H:%M:%S", tm_info);
  sprintf(buffer, "%s.%d", buffer, millisec);
}

int main(int argc, char* argv[]) {
        if (argc!=1) {
                printUsage();
                exit(1);
                return 1;
        }

        init();

        pthread_t *thread;
        long threadid;
        int rc;
        rc = pthread_create(&thread, NULL, countdown, (void *)threadid);
        if (rc){
                fprintf(stderr,"ERROR; return code from pthread_create() is %d\n", rc);
                return(1);
                exit(1);
        }

        while (1) {
                delay(LOOP_DELAY);
                if (DEBUG) fprintf(stdout, "%d ",timer);
                if (DEBUG) fflush(stdout);

                if (timer==0) {
                        display_off();
                        while (!timer) {
                                delay(1000);
                        }
                        display_on();
                }
        }

        exit(EXIT_SUCCESS);
        return 0;
}
```

ANEXO II

```
modules:[
{
 module: 'camera',
 position: 'top-center',
 config: {
        selfieInterval: 3,
        emailconfig: {
                service: 'Hotmail',
                auth: {
                    user: '<stefanyaquino@hotmail.com>'
                    pass: '<*********>'
                }
        }
 }
},
{
 module: 'voicecontrol',
 position: 'bottom-left',
config: {
        models: [
                {
                keyword: "Show Camera",
                description: "Diga 'Mostrar Camera' para utilizar a camera.",
                file: "mostrarcamera.pmdl",
                message: "SHOW_CAMERA"
                },
                {
                keyword: "Hide Camera",
                description: "Diga 'Desligar Camera' para fechar a camera.",
                file: "desligarcamera.pmdl",
                message: "HIDE_CAMERA"
                },
                {
                keyword: "Selfie",
                description: "Diga 'Selfie' para tirar foto.",
                file: "selfie.pmdl",
                message: "SELFIE"
                },
                ]
        }

},

]
```