

Sistemas Embarcados

Apresentação final:

Interactive self-esteem mirror Lu² (Isem Lu²)

Luiza Irina Lima dos Santos

Universidade de Brasília

Faculdade Gama - FGA

Brasília, Brasil

luizairinalds@gmail.com

Luciana Alves Fernandes

Universidade de Brasília

Faculdade Gama - FGA

Brasília, Brasil

lucianaaf12@gmail.com

Resumo— *A Revolução Digital permitiu a introdução de inovações como a Internet das coisas, a qual pode ser interpretada como a capacidade de objetos transmitirem dados para uma rede e realizarem de modo incisivo a interação objeto-usuário. Nesse sentido, para uma maior eficiência no dia a dia com um toque de estímulos, esse projeto baseia-se na criação de uma tela espelhada interativa que auxiliará tanto na otimização do tempo quanto na auto-estima do usuário.*

Palavras-chaves— *sistema; espelho; Raspberry; auto-estima;*

1. INTRODUÇÃO

A tecnologia está sempre em constante desenvolvimento e aprimoramento e suas aplicações expandiram de forma significativa e espantosa. Com a atual facilidade de acesso a internet mecanismos como a Revolução Digital permitiu que essa ficasse intrínseca a vida de um indivíduo. Basta olhar ao redor para verificar que o mundo está permeado de ferramentas que visam uma maior interação com o ser humano, auxiliando-o a realizar suas atividades do dia-a-dia.

1.1. REVISÃO BIBLIOGRÁFICA

A sociedade do século XXI, em questão, tem como problemas pontuais a chamada falta de tempo e baixa auto-estima, isso inclui de maneira sensível o acúmulo de responsabilidades e consequentemente o fato de agregar fatores do intelecto quanto a sua forma de ser e estar, o que pode gerar tanto a baixa produtividade com relação as tarefas habituais (ou não) quanto o surgimento de esquecimentos devido ao estresse. O conceito da Revolução Digital carrega consigo uma mudança relativa nos modos de vida do ser humano como solução revolucionária, introduzindo de forma abrangente e diferenciada as definições do termo Internet das Coisas. A Agência Brasileira de Internet das Coisas (abinc) diz que essa

tecnologia pode ser cogitada como a Terceira Geração da Internet [2].

O termo ‘Internet das coisas’, por fazer parte de aplicações expansivas, pode ser interpretada como a capacidade de objetos, no caso desse projeto, realizar a transmissão de dados para uma rede proporcionando interação entre objeto-usuário [3]. Ou seja, antes o que era um televisor, o qual apenas transmitia respectivos canais, atualmente com esse engendramento tem-se a possibilidade do televisor ter mais funcionalidades, como a conexão na rede Wi-Fi.

Segundo Lemos (Lemos, André, 2012), os objetos dispõem de abundantes funcionalidades e entre elas têm-se o reporte de lembranças, sentimentos e costumes predominantes. Desde os primórdios da humanidade, os utensílios vêm mudando de modo recíproco a acompanhar as novas experiências e descobertas dos seus usuários. Com a introdução da inovação da Internet das coisas, tem-se a possibilidade de tornar um objeto comum ao cotidiano- no projeto tratado um suposto espelho- interativo e auxiliador tanto na otimização de tempo funcional como também na elevação da auto-estima para uma maior produção diária, tendo em vistas as atividades pessoais de cada ser.

1.2. JUSTIFICATIVA

O projeto *Interactive self-esteem mirror Lu²* foi motivado por permitir a evolução de objetos, que outrora eram simples utensílios, adaptando-os as tecnologias inovadoras, como a aplicação da Internet das coisas. O uso da Raspberry é primordial, pois possibilita a criação de um sistema embarcado para a aplicação específica em questão. Isso não seria possível ao usar um *smartphone*, por exemplo, vários aplicativos e funções do mesmo ficariam inutilizáveis, o que pode ser visto como um prejuízo perante a tecnologia, a qual é de propósito geral.

Para entender melhor a aplicação proposta foi criado um estudo de caso, em que pode-se perceber a utilidade do espelho e sua praticidade com a adição da tecnologia no cotidiando da Ana.

Ana acordou animada para seu dia de trabalho, e como de costume, foi até o espelho para se vestir.



Mas então, perguntas começaram a surgir na cabeça de Ana. "Será que a reunião com o chefe era hoje? E o almoço com Cristina, eu remarquei? E será que fará frio ou calor?"



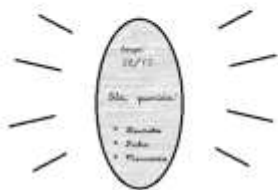
E agora, Ana, que roupa usar?



Nesse momento, Ana, através de um simples comando de voz, ativou o espelho mágico.



Ela viu que seria um dia quente, e também leu sua lista de atividades. Assim, Ana pode verificar que sua reunião estava marcada para aquele dia e que não teria almoço com sua amiga.



Tem que ser esta!



1.3. OBJETIVOS

Com o intuito de maior praticidade no dia a dia e maior interatividade para com o usuário, esse projeto baseia-se na criação de uma tela espelhada interativa *Interactive self-esteem mirror Lu² (Isem Lu²)*, que forneça dados como a data (calendário), hora, temperatura, atividades do dia e mensagens de áudio inspiradoras e alegres. O sistema construído seria um elevador de autoestima, embarcado pela Raspberry-Pi.

1.4. REQUISITOS

Os requisitos do projeto podem ser descritos em duas partes, as quais tratam dos aspectos do *Hardware* e do *Software*.

- Aspectos do *Hardware*:

- Raspberry Pi;
- Cartão SD, para a memória do sistema;
- O televisor tem de ser de Led;
- Aspecto espalhado: acrílico espelhado;
- Conexão HDMI;
- Conexão internet;
- Sensor de presença;

-Aspectos de *Software*:

- Biblioteca para comando de voz;
- Sistema operacional embutido na placa *Raspberry*;
- Programação dos módulos específicos;
- Programação da módulo do espelho;
- *Script* para a execução de arquivos;

1.5. BENEFÍCIOS

O projeto do espelho inteligente aparentemente parece ter um leque restrito de benfeitorias. Porém, esse pensamento é efetivamente errôneo. Basta analisar o cotidiano de um indivíduo, enquanto a pessoa se arruma na frente do espelho para iniciar sua rotina, o dispositivo apresenta diversas possibilidades de uso para um simples espelho. O fator atrativo infere-se na interação sistema-usuário, o que acentua informações úteis ao indivíduo, mostrando-lhe as horas, a data, bem como a agenda de atividades de maneira prática, listando as metas do dia e também, executando mensagens motivacionais para que o indivíduo se sinta determinado a concretizar seus afazeres. Essas atividades não precisam tratar apenas de reuniões de trabalho, plano de estudo da faculdade, pode ser um lembrete aos idosos, por exemplo, para que eles não esqueçam de tomar um remédio de pressão, entre inúmeras aplicações de acordo com a rotina do usuário.



Figura 1. Ilustração do espelho mágico. Fonte: <https://hackaday.io/project/13466/gallery#da49a892f905e0eb285052618d1ef784>, 2017.

2. DESENVOLVIMENTO

As descrições apresentam as etapas do projeto e seu desenvolvimento atual com seus determinados detalhes.

2.1. DESCRIÇÃO DO HARDWARE

O projeto é composto 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.

Na frente do monitor, uma tela de acrílico espelhado deveria ser inserida, essa será a responsável pela aparência de espelho do dispositivo inteligente a ser desenvolvido. No entanto o custo da tela de acrílico espelhado ficou fora do alcance dos integrantes de modo que outra solução foi encontrada, comprou-se uma chapa de acrílico normal e uma película espelhada foi inserida no material de maneira a apresentar o mesmo efeito esperado do acrílico espelhado.

Também será utilizado um microfone, que nesse caso precisa ser USB para ativação do comando de voz e como saídas de áudio será utilizado o autofalante da própria TV.

Para que o espelho seja ligado será usado para monitorar a presença do usuário um sensor PIR, o qual apresenta saída no modo digital, não exigindo a necessidade de um conversor A/D. Dessa maneira, pode-se economizar energia e fazer uma interação direta com o usuário que se aproxima.

A) Lista de Materiais:

Para a realização deste projeto foram necessários os componentes descritos no Quadro 1.0.

Quadro1.0. Lista de Materiais utilizados.

Quantidade	Material
1	TV-LED
1	Sensor PIR
1	Placa de acrílico
1	Película espelhada
1	Raspberry Pi 3
-	Jumper
1	Microfone USB
1	Protoboard pequena

B) Monitor TV-LED + Acrílico espelhado:

Para que o aspecto de espelho ficasse conforme esperado foi necessário comprar uma placa de acrílico transparente e aplicar nesta a película espelhada para ter o efeito translúcido gerando a ideia do espelho.

C) Microfone:

Para a aplicação necessitou-se de um microfone USB, devido a atuação da Raspberry. Para o mesmo funcionar teve de configurá-lo para que a Raspberry o reconhecesse. Primeiramente, a Raspberry foi conectada via ssh e cabo serial e logo depois foi digitado o seguinte comando no terminal:

```
$ lsusb
```

Esse comando permitiu verificar que o microfone foi reconhecido pelo terminal USB da Raspberry. O próximo passo foi configurar a escolha e o ganho do microfone, para isso utilizou-se o comando:

```
$ alsamixer
```

O comando abordado acima abre a janela de configuração, apertando-se F6 escolhe-se a entrada respectiva do microfone. O ganho foi definido ao apertar em F4, e foi ajustado no máximo permitido. Feitas as configurações citadas o microfone foi testado com os seguintes comandos no terminal em que como o microfone estava setado como 1, o plughw apresenta o 1 como saída escolhida. Nesse caso, um trecho que uma música foi cantado e utilizando a interrupção CTRL+C encerrou-se a gravação.

```
$ arecord -D plughw:1,0 teste.wav
```

Para ouvir o áudio gravado no arquivo chamado teste.wav digitou-se no terminal o comando abaixo, o que permitiu escutar o trecho da música que foi gravada.

```
$ aplay -D plughw:0,0 teste.wav
```



Figura 2. Microfone USB utilizado.

D) Sensor de presença:

O sensor de presença utilizado foi o PIR (Passive infrared sensor) HC-SR501. Este 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 3.0 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 3 como “*Distance Adjust*”, quanto mais no sentido horário, maior será a sensibilidade adquirida.

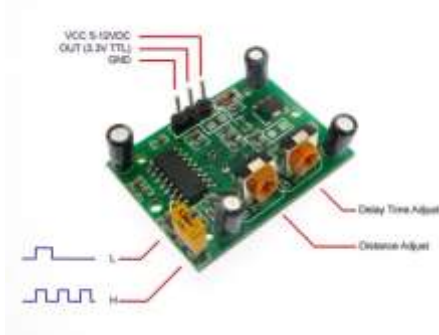


Figura 3. Visão inferior do sensor PIR.

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 conectado 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.

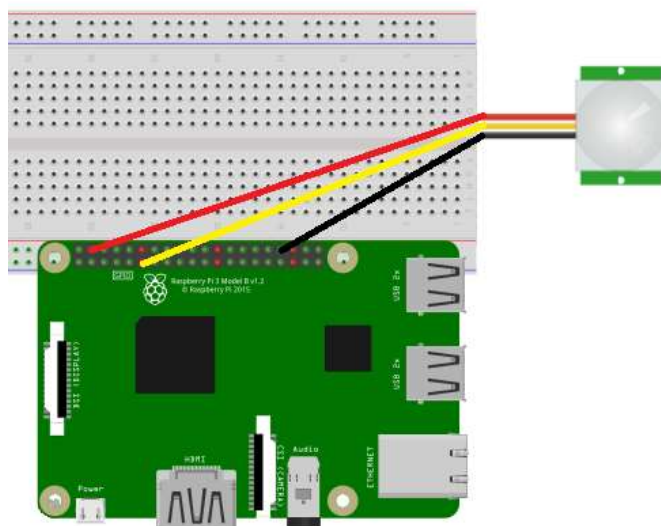


Figura 4. Esquemático do hardware-Sensor PIR .

2.2. DESCRIÇÃO DO SOFTWARE

O sistema operacional embutido na placa raspberry terá como funcionalidade gerenciar a ordem de execução das partes do *software*. Ele será responsável pela comunicação entre a inteligência do projeto e os dispositivos de saída e entrada de dados, como os dados apresentados na tela, o microfone e o autôfalante da televisão.

A Figura 5 apresenta o diagrama de blocos do sistema como um todo. Observa-se que se tem o fluxo de

estados e ao chegar no estado final observa-se as funcionalidades do espelho. O acionamento ocorrerá por meio da atuação do sensor PIR mais comando de voz, com a biblioteca *Pocketsphinx* no momento em que o usuário mencionar alguma palavra-chave.

Entre os módulos específicos do sistema, compreende-se o acionamento das mensagens motivacionais, em que necessita-se de uma biblioteca de comando de voz, a *espeak*. O espelho inteligente conectará a internet para mostrar na tela a agenda de atividades atualizada do usuário, assim como outras informações que se queira aderir.

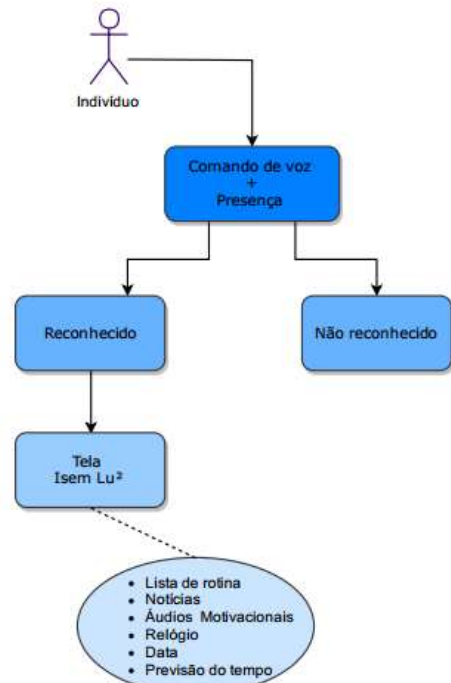


Figura 5. Diagrama de blocos do sistema Isem Lu².

A) Sistema Operacional da Raspberry:

O primeiro passo de todo o projeto foi a escolha do sistema Operacional a ser utilizado. Fez-se uma pesquisa e escolheu-se o Sistema Operacional *Raspbian*, o qual mostrou-se completo e requisitado por iniciantes justamente por permite um fácil manuseio e ter em sua disposição ferramentas de configuração do computador. O sistema *Raspbian* e o instalador *Noobs* foram inseridos no cartão SD para realizar o processo de instalação. Feito isso, a forma de acesso remoto teve de ser escolhida, para esse caso escolheu-se a SSH (*Secure Shell*), a qual possibilita uma atuação com uma maior segurança. O próprio sistema *Raspbian* permitiu selecionar SSH e escolher a opção de enable.

B) Código para o sensor PIR:

O código em Anexos que generaliza o projeto tem como objetivo ligar o sistema a partir do sinal do sensor de presença PIR. Quando o sensor detecta um objeto em sua

proximidade, sua saída digital vai de nível lógico baixo para alto. Para comunicação entre o sensor PIR e a *Raspberry* foi usada a biblioteca *Wiring PI*, que interfacea os pinos GPIO da *Raspberry*. A função usada foi a "wiringPiISR", que permite uma interrupção por um pino GPIO. A borda de subida do sinal digital do sensor ativa a função "mov_now". Nessa função um aviso de movimento é enviado e o sistema do espelho mágico é inicializado pelo script chamado pela função "system". Ainda na função da interrupção do sensor, uma variável recebe o tempo em que o sistema foi ativado.

No *loop* infinito uma verificação de tempo é feita para desligar o espelho mágico em cinco minutos caso não tenha havido nenhuma detecção de presença pelo sensor PIR. Como o sensor detecta constantemente alguém em sua frente, a variável "a" faz o controle para o *script* de inicialização não ser chamado repetidamente pelo "system".

C) Módulos do espelho:

O projeto do espelho inteligente possui um código principal que fica executando continuamente. Essa é a parte que controla a interface gráfica mostrada na tela do espelho. O código principal é dividido em módulos que são os 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. Porém, os módulos que são padrões para que funcionem corretamente de acordo com a localidade em que o espelho irá atuar precisam ser configurados de acordo com o modo de vida do usuário. No caso do espelho inteligente de auto-estima os módulos a serem acrescentados foi o calendário ressaltando a lista de atividades do usuário e as mensagens de frases incentivadoras. O *script* de reprodução de áudio com as frases e o reconhecimento de voz será agregado ao código principal do sistema. A Figura 6 apresenta a imagem do espelho com os padrões sem as configurações e alterações necessárias.



Figura 6. Interface gráfica do projeto base MagicMirror implementado na Raspberry.

O programa principal do Isem Lu² tem as

configurações de variáveis iniciais e as chamadas dos módulos com os parâmetros de entrada. Os módulos funcionam como funções que estão descritas em arquivos dentro de um diretório próprio para módulos na pasta do projeto. Em cada arquivo de módulo tem-se toda a lógica do aplicativo, bem como sua conexão com a internet se for o caso, e seus modos de operação de acordo com os parâmetros de entrada. Novas funções devem ser colocadas nesse formato de módulo para se tornarem uma aplicação do espelho inteligente.

O programa principal, "config.js", é chamado com um *script* pelo terminal por meio do comando 'pm2', que inicia um processo *daemon* e faz a execução do programa chamado com um *fork*. O 'pm2' pode ser chamado para monitorar a execução do programa do espelho inteligente e também pará-lo. Vale ressaltar que é no arquivo do programa principal que são configurados os módulos e acrescentados os específicos. Abaixo serão descritos cada módulo utilizado atualmente no Isem Lu².

- **Módulo da Lista de atividades:**
Trata-se do módulo específico que contempla a descrição da rotina do indivíduo. Esta rotina é inicialmente feita na agenda do *gmail* do usuário e é atualizada no espelho a cada cinco minutos, verificando possíveis modificações.
- **Módulo da hora e data:**
Trata-se da configuração da hora com as características dos parâmetros definidos pela localidade, no caso desse projeto o local definido foi a cidade de Brasília-BR. Esse módulo permite escrever por extenso o dia da semana e o numeral da data correspondente.
- **Módulo da previsão do tempo:**
Trata-se de dois módulos, o primeiro apresenta apenas o valor da temperatura atual em graus *Celsius*, porém caso o usuário queira, este permite que acrescente a velocidade do vento e a direção. Para esse projeto retirou-se a última descrição. O segundo módulo apresenta a previsão do tempo tendo em vista mais dias da semana com as previsões de temperaturas máximas e mínimas, o que permite ao usuário uma visibilidade maior para poder se planejar e se arrumar de acordo com as informações apresentadas.
- **Módulo de notícias:**
Trata-se dos resumos dos noticiários para que o usuário ao arrumar antes de sair de casa já fique informado quanto aos eventos mais atuais. Para apresentar o espelho foi configurado esse módulo para mostrar as notícias do G1.
- **Módulo das frases:**
O módulo das frases foi retirado nesse ponto de

controle. Pois, o intuito agora foi definido para as frases faladas utilizando a biblioteca *espeak*.

D) Biblioteca *Espeak*:

A biblioteca *espeak* permite realizar a conversão text-to-speech(TTS), em que um texto é sintetizado para voz. Essa biblioteca foi escolhida por ser de fácil manipulação, além de corresponder com as expectativas destinadas a mesma e por conter em sua base a plataforma em português. A biblioteca apresenta limitações principalmente se tratando de uma voz que não parece ser natural. Entretanto a biblioteca permite o uso tanto de uma voz masculina como feminina e estas podem ser reguladas por intensidade, além de poder reproduzir arquivos em .wav. A biblioteca foi testada por enquanto apenas no terminal para teste. Para isso, basta instalar a mesma e digitar o comando para obter a saída que pode ser ouvida em um fone de ouvido inserido na *Raspberry*. O “-vpt + f2” trata-se da chamada em português com voz feminina com uma intensidade dois. Essa intensidade quanto maior percebeu-se que a voz aparentava mais grossa.

Comando:

```
$espeak -vpt + f2 “Seja feliz sempre”
```

A possibilidade de uso dessa biblioteca permite criar arquivo .txt, como o apresentado na Figura 7, com o que se deseja falar e utilizando as configurações pode-se fazer uma chamada se sistema para a execução do comando do sintetizador de voz. A chamada do sistema pode ser observada no segundo código em Anexo, sendo este um código de teste apenas para a biblioteca *espeak*. A opção -vpt-br habilita a linguagem para português do Brasil. O ajuste -k trata-se de um tratamento para as letras maiúsculas que iniciam as frases, permitindo ressaltar estas usado nesse caso o valor -20. O -f indica que um arquivo será lido.

```
" Ola bom dia"  
" Que voce seja muito feliz "  
" O sol hoje esta radiante"
```

Figura 7. Imagem do arquivo teste.txt para testar a biblioteca *espeak*.

E) Biblioteca *Pocketsphinx* (*speech-to-text*):

Inicialmente para trabalhar com o reconhecimento de voz foi estudada algumas bibliotecas e a escolhida foi a *CMUSphinx*, a qual é livre para ser baixada e permite que a atuação da mesma funcione sem a necessidade da internet. A biblioteca *CMUSphinx* é uma biblioteca que abrange, no caso desse projeto, a *Pocketsphinx*. Esta oferece funcionalidade perante ao interesse de Reconhecimento de voz.

A biblioteca citada foi encontrada disponível em *python*. A chamada foi realizada usando o comando

continuos, o que possibilitou uma maior eficiência da obtenção de dados. A biblioteca *Pocketsphinx* é a mais importante em vista nesse projeto, pois trata-se da conversão *speech-to-text*, fornece recursos em que ao executar o *script* de teste, a biblioteca pergunta o que você tem a dizer e escreve na tela o que o usuário disse, esse seria o teste inicial.

O microfone USB foi plugado antes mesmo da instalação, pois isso é o recomendado para que a instalação ocorra com sucesso. Vale ressaltar que a *Pocketsphinx* não possui boa acurácia perante sua funcionalidade, para aumentar esta foi necessário criar um vocabulário para delimitar as palavras do conhecimento da biblioteca. Isso permite que a mesma escreva com maior exatidão o que foi escrito, entretanto não infere que os erros foram completamente resolvidos.

A Figura 8 apresenta o vocabulário para o Isem Lu². Para que a biblioteca reconheça as palavras contidas no vocabulário foi inserido o arquivo .txt em um site educacional e este retorna um arquivo zipado, o qual é composto por dois arquivos com extensão .lm e .dic, estes é quem irão permitir que a biblioteca reconheça o que foi falado e escreva de acordo com o vocabulário determinado. A chamada da biblioteca é feita através do seguinte comando:

```
$ pocketsphinx_continuous -adcdev plughw:1,0 -lm  
/home/pi/9855.lm -dict /home/pi/9855.dic -inmic yes
```

```
Magic  
stop  
good  
legal  
morning  
nice  
beautiful  
cool  
today  
night  
afternoon  
darling  
start  
pause
```

Figura 8. Imagem do vocabulário limitado para a biblioteca *Pocketsphin*.

F) Código do sistema Isem Lu²:

Com todos os blocos funcionando de forma separada foi possível engendrar o código acoplando todo o sistema. Estudou-se como organizar todo o sistema de modo que as funcionalidades trabalhassem paralelamente. Com o conhecimento adquirido durante as aulas escolheu-se implementar o conjunto com as *Threads*. Estas são mecanismos que possibilitam um programa operem mais de uma função de modo simultâneo. Além dessa vantagem, tem-se também o compartilhamento de memória tornando a comunicação do programa mais simples. Comitantemente, houve a preocupação de que todas as *Threads* deveriam rodar, pois o sistema operacional agenda-as assincronamente não diferenciando a

ordem de execução. Para resolver essa questão de sincronismo usou-se os conceitos de *Mutex* (*Mutual Exclusive*).

O *Mutex* é uma possível solução para que as *Threads* rodem em sincronismo. Uma variável é compartilhada e para que funcione corretamente as ações do *Mutex* devem ser executadas, as quais se tratam de adquirir ou liberar a chave de acesso. Vale ressaltar que se a chave estiver sendo usada tem-se uma ação bloqueante.

Voltando ao código, apresentado no Anexo 3, tem-se inicialmente as declarações das bibliotecas, as quais permitem realizar comunicação de dados, usar as *Threads*, funções de tempo, sinais, entre outras. Logo depois, declarou-se o *Mutex* e as *Threads*. A primeira foi criada para que a *pocketsphinx* atuasse de maneira constante. O que a biblioteca escutou e o retorno dado ao usuário foram armazenados no arquivo “resposta.txt”, este possibilitou acionar comandos da *espeak*, os quais serão descritos mais adiante.

A segunda *Thread* foi criada para atuar no sensor, pois o mesmo deve funcionar continuamente para detectar ou não movimento e consequentemente ligar ou desligar o espelho. O pino habilitado como entrada digital definido para o sensor foi o 11 da *Raspberry*, utilizou-se os pinos de 5V e de *ground*. A comunicação usada entre o sensor PIR e a *Raspberry* foi a serial, em que ao detectar movimento encaminha-se para a função “move_now”. O *loop* infinito verifica após o reconhecimento de movimento se este passou de 30 segundos. Ao passar 30 segundos sem detecção de movimento é realizado a chamada *system* para desligar o espelho. A função *move_now* apresenta na tela o reconhecimento do movimento e habilita a chamada *system* para ligar o espelho. Duas *flags* foram inseridas para uma facilitar a visualização do espelho nos modos ligado e desligado.

A função *main* inicia-se com a declaração do ponteiro do arquivo, *Threads* e demais variáveis utilizadas durante o código. As *Threads* foram criadas. Para que utilizar o reconhecimento de voz para a finalidade de emitir frases motivacionais abriu-se o arquivo “resposta.txt” para leitura e em uma estrutura de repetição ao ter a variável *x* igual a zero o arquivo foi lido e a palavra padrão era armazenada em um buffer. O *Mutex* auxiliou no sincronismo, em que ao falar a palavra padrão e consequentemente ao reconhecer logo após a biblioteca *espeak* era acionada para dizer a frase. Nota-se que a *espeak* foi configurada com parâmetros para melhorar a pronuncia e o som das palavras. A *espeak* foi configurada para falar português do Brasil, o tom definiu-se em 80, para essa frase colocou-se um volume de 200, o qual é o máximo e o ajuste de velocidade para pronunciar a frase foi de 150, o *g* também é um parâmetro de tonalidade e o *k* é para intensificar a voz a cada vez que uma letra maiúscula for identificada. Após a execução da *espeak* o *Mutex* liberou o acesso. Foram usadas como palavras-chaves: *Magic*, *Beautiful*, *Morning*, *Afternoon* e *Darling*.

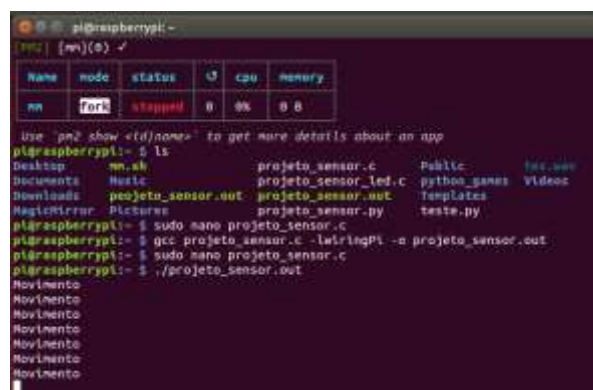
Ainda na função *main*, para que a mesma ficasse ativada até todas as *Threads* rodarem utilizou-se a função “*pthread_join()*”. Esta função possui dois argumentos, os

quais são a *thread_ID* que esperará e o ponteiro *void**, este receberá o valor de retorno da *thread*.

3. RESULTADOS

A partir de todo o desenvolvimento pode-se realizar alguns testes iniciais com todo o sistema em funcionamento. Vale ressaltar que o sistema ainda está em andamento de melhorias, o que não permitiu apresentar um resultado teste do sistema Isem Lu² como um todo.

A Figura 9 e 10 apresentam o resultado teste para a atuação do sensor PIR, que é responsável por detectar movimento a partir da variação da luz infravermelha, no terminal e no circuito montado, respectivamente.



```
pi@raspberrypi:~$ ./projeto_sensor.out
Name      mode      status    CPU    memory
nm        Fork      Stopped   0      0%      0 B

Use 'nm show <id/name>' to get more details about an app

pi@raspberrypi:~$ ls
Desktop  nm.sh      projeto_sensor.c  Public:  test.wav
Documents  Hostile    projeto_sensor_led.c  python_games  Videos
Downloads  projeto_sensor.out  projeto_sensor.c  templates
Music       Pictures   projeto_sensor.py  teste.py
pi@raspberrypi:~$ sudo nano projeto_sensor.c
pi@raspberrypi:~$ gcc projeto_sensor.c -luringPI -o projeto_sensor.out
pi@raspberrypi:~$ sudo nano projeto_sensor.c
pi@raspberrypi:~$ ./projeto_sensor.out
Movimento
Movimento
Movimento
Movimento
Movimento
Movimento
```

Figura 9. Teste realizado para a detecção do movimento.



Figura 10. Circuito do sensor PIR montado.

A Figura 11 e 12 apresentam a parte do hardware que traz embutido a tela inicial do Isem Lu², a primeira figura apresenta a tela do Isem Lu² configurada, com a lista de atividade de um usuário e as demais configurações personalizadas para o usuário. Nota-se, na Figura 12 que realmente tem-se a impressão de estar em frente a um espelho, o que foi possibilitado devido o efeito translúcido do acrílico.

economia de energia de parte do sistema. A biblioteca de de speech-to-text funcionou como esperado dentro do vocabulário delimitado assim como a biblioteca *tex-to-speech*.

O sistema como um todo funcionou de modo satisfatório, o engedramento do código do sistema *Isem Lu²* permitiu colocar em prática a teoria transmitida em sala de aula abrangendo desde o início da matéria com conceitos de abertura de arquivo, leitura de arquivo texto e inclusive os pontos abordados ao final como *Threads*, *Mutex*, comunicação serial.

Observa-se que o projeto possibilitou uma aplicação diferenciada por se tratar da interferência na auto-estima do usuário através das frases motivacionais e da facilidade de visão das informações importantes que englobam mais de um dia. Ou seja, antes de sair de casa pode-se verificar, por exemplo, se o dia será frio ou não e compor uma vestimenta de acordo com o que foi visto.

A execução do projeto em uma visão geral foi proveitosa principalmente por inserir os alunos em uma problemática por eles abordada e sujeitos a problemas inesperados, o que fez com que fosse trabalhado não somente os conceitos aplicados como também a postura de se lidar com as dificuldades como a busca por solucionar os erros e acrescentar possíveis melhorias.

5. REFERÊNCIAS BIBLIOGRÁFICAS

- [1] Teeuw, Michael, Magic Mirror: Part I – The Idea & The Mirror:
< <http://michaeltew.nl/post/80391333672/magic-mirror-part-i-the-idea-the-mirror> > Acessado em 01 de Abril de 2017.

- [2] 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 03 de Abril de 2017.

- [3] Yoshida, Hubert, Encontro na Poli discute futuro da internet das coisas, Publicado em Tecnologia, USP Online Destaque por Redação em 25 de abril de 2014:

<<http://www5.usp.br/42858/encontro-na-poli-discute-futuro-da-internet-das-coisas/>> Acessado em 03 de Abril de 2017.

- [4] Lemos, André, A comunicação das coisas. Internet das Coisas e Teoria Ator-Rede: Etiquetas de radiofrequência em uniformes escolares na Bahia, Salvador- Brasil, 2012.

<http://s3.amazonaws.com/academia.edu.documents/36911184/Andre_Lemos.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1491316199&Signature=q39qJqELSS3rVtrwGxfthcJbmVU%3D&response-content-disposition=inline%3B%20filename%3DAndre_Lemos.pdf> Acessado em 03 de Abril de 2017.

- [5] Singer, Talyta, TUDO CONECTADO: CONCEITOS E REPRESENTAÇÕES DA INTERNET DAS COISAS, Salvador-Brasil, 2012:

<<http://www.simsocial2012.ufba.br/modulos/submissao/Upload/44965.pdf>> Acessado em 03 de Abril de 2017.

- [6] Raspberry Pi Smart Mirror:

<<https://hackaday.io/project/13466-raspberry-pi-smart-mirror>> Acessado em 03 de Abril de 2017.

- [7] Garcia, Diogo: Aulas de Sistemas Embarcados:

<https://github.com/DiogoCaetanoGarcia/Sistemas_Embarcados/blob/master/Aulas/06_Recursos%20do%20Sistema%20Linux%20-%20Parte%206%20-%20Threads.pdf> Acessado em 28 de julho de 2017.

- [8] Cardoso, Bruno Lima, SISTEMA DE RECONHECIMENTO DE FALA BASEADO EM POCKETSPHINX PARA ACESSIBILIDADE EM ANDROID, Universidade Federal do Rio de Janeiro- Rio de Janeiro, 2015.

Anexos

Anexo 1. Código de teste para a detecção de presença e ativação da tela Isem Lu² :

```
[2]  # include <stdio.h>
[3]  # include <stdlib.h>
[4]  # include <pthread.h>
[5]  #include <unistd.h>
[6]  #include <string.h>
[7]  #include <wiringPi.h>
[8]  #include <stdio.h>
[9]  #include <sys/types.h>
[10] #include <signal.h>
[11] #include <pthread.h>
[12] #include <stdlib.h>
[13] #include <time.h>
[14]
[15] char a=0;
[16] clock_t tInicio, tFim, tDecorrido;
[17]
[18] void move_now(void){
[19]     fprintf(stderr,"Movimento\n");
[20]     if(a==0){
[21]         system("pm2 start mm.sh");
[22]         a=1;
[23]     }
[24]     tInicio = clock();
[25] }
[26]
[27] int main(){
[28]
[29] int pin_PIR = 11;
[30]
[31] wiringPiSetup();
[32]
[33] pinMode(pin_PIR,INPUT);
[34]
[35] wiringPiISR(0,INT_EDGE_RISING, move_now);
[36]
[37] while(1) {
[38]     tFim = clock();
[39]     tDecorrido = ((tFim - tInicio) / (CLOCKS_PER_SEC / 1000));
[40]
[41]     //300000 ms = 5 min
[42]     if (tDecorrido>300000){
[43]         system("pm2 stop mm");
[44]         a==0;
[45]     }
[46]
[47] }
[48] pthread_exit(NULL);
[49] return 0;
[50] }
```

Anexo 2. Código de teste da biblioteca espeak com arquivo:

```

[51] //include <wiringPi.h>
[52] #include <stdio.h>
[53] #include <sys/types.h>
[54] #include <pthread.h>
[55] #include <stdlib.h>
[56] #include <time.h>
[57]
[58] int main() {
[59]     system("espeak -vpt-br -k -20 -f teste.txt");
[60]     return 0;
[61] }

```

Anexo 3. Código de todo o sistema Isem Lu²:

```

[62] # include <stdio.h>
[63] # include <stdlib.h>
[64] # include <pthread.h>
[65] #include <unistd.h>
[66] #include <string.h>
[67] #include <wiringPi.h>
[68] #include <sys/types.h>
[69] #include <signal.h>
[70] #include <time.h>
[71]
[72] static pthread_mutex_t mutexLock;
[73]
[74] //void *thread_function(void *arg);
[75] void *thread_escuta(void *arg);
[76] void *thread_sensor(void *arg);
[77]
[78]
[79] void move_now(void);
[80]
[81] char a=0, b=1;
[82] clock_t tInicio, tFim, tDecorrido;
[83]
[84] int main() {
[85]     int res;
[86]     pthread_t a_thread, s_thread;
[87]     void *thread_result;
[88]
[89]     // inicio o MUTEX
[90]     res = pthread_mutex_init(&mutexLock, NULL);
[91]     res = pthread_create(&s_thread, NULL, thread_sensor, NULL);
[92]     if (res != 0) {
[93]         perror("Não foi possível criar a thread!");
[94]         exit(EXIT_FAILURE);
[95]     }
[96]
[97]     res = pthread_create(&a_thread, NULL, thread_escuta, NULL);
[98]     if (res != 0) {
[99]         perror("Não foi possível criar a thread!");
[100]         exit(EXIT_FAILURE);
[101]     }
[102]     sleep(5);
[103]
[104]     FILE *f;
[105]     char buffer[1026];
[106]     char padrao[6] = "MAGIC";

```

```

[107] int x = 0;
[108]
[109] //Abrindo arquivo para a pocket:
[110] f = fopen("resposta.txt","r");
[111] if(f == NULL){
[112]     perror("The following error occurred");
[113]     return;
[114] }
[115] while(x==0) {
[116]
[117]     if( fgets(buffer,1025,f) != NULL ){
[118]         if( strstr(buffer,padrao) != NULL ){
[119]             printf("pronto, escutei voce \n");
[120]             pthread_mutex_lock(&mutexLock);
[121]             system("espeak -vpt-br+f5 -p 80 -a 200 -s 150 -g 10 -k 20 'Ola gente legal bonita e maravilhosa' ");
[122]             pthread_mutex_unlock(&mutexLock);
[123]             x=1;
[124]
[125]             // dou uma pausa na thread ao inves de cancelar$
[126]             //pthread_mutex_lock(&mutexLock);
[127]             //fclose(f);
[128]             //system("rm resposta.txt");
[129]             //f = fopen("resposta.txt","rw");
[130]             x=0;
[131]             //pthread_mutex_unlock(&mutexLock);
[132]             sleep(1);
[133]             puts(buffer);
[134]         }
[135]     else {
[136]         char palavrinha[] = "BEAUTIFUL";
[137]         char querida[] = "DARLING";
[138]         char hoje[] = "TODAY";
[139]         char tarde[] = "AFTERNOON";
[140]         char manha[] = "MORNING";
[141]
[142]         if( strstr(buffer,manha) != NULL ){
[143]             pthread_mutex_lock(&mutexLock);
[144]             system("espeak -s 115 -v pt-br+f4 'Bom dia que hoje seja excelente' ");
[145]             pthread_mutex_unlock(&mutexLock);
[146]         }
[147]
[148]         if( strstr(buffer, tarde) != NULL ){
[149]             pthread_mutex_lock(&mutexLock);
[150]             system("espeak -s 115 -v pt-br+f4 'Boa tarde Acredite em você' ");
[151]             pthread_mutex_unlock(&mutexLock);
[152]         }
[153]
[154]         if( strstr(buffer,palavrinha) != NULL ){
[155]             pthread_mutex_lock(&mutexLock);
[156]             system("espeak -s 115 -v pt-br+f4 'Linda você é radiante' ");
[157]             pthread_mutex_unlock(&mutexLock);
[158]         }
[159]
[160]         if( strstr(buffer, querida) != NULL ){
[161]             pthread_mutex_lock(&mutexLock);
[162]             system("espeak -s 115 -v pt-br+f4 'Querida Faça o seu melhor sempre' ");
[163]             pthread_mutex_unlock(&mutexLock);
[164]         }
[165]         puts(buffer);
[166]     }

```



```

[165]
[166]     }
[167] }
[168] fclose(f);
[169]
[170] printf("Esperando o fim da execução da thread ... \n");
[171] res = pthread_join(a_thread,&thread_result); //thread_result é o pontei$
[172] res = pthread_join(s_thread,NULL);
[173] if (res != 0){
[174]     perror("Não foi possível juntar as threads!");
[175]     exit(EXIT_FAILURE);
[176] } exit(EXIT_SUCCESS);
[177] }
[178]
[179] void *thread_escuta(void *arg){
[180]     int i, res;
[181]     res = pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);
[182]     if (res != 0){
[183]         perror("Falha na pthread_setcancelstate");
[184]         exit(EXIT_FAILURE);
[185]     }
[186]     res = pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED, NULL);
[187]     if (res != 0){
[188]         perror("Falha na pthread_setcanceltype");
[189]         exit(EXIT_FAILURE);
[190]     }
[191]     //printf("Função thread executando. \n");
[192]     system("pocketsphinx_continuous -adcddev plughw:1,0 -lm /home/pi/9855.lm -dict /home/pi/9855.dic -inmic yes >> resposta.txt");
[193]
[194]     /*for (i = 0; i < 10; i++){
[195]         printf("Thread em execução (%d) ... \n", i);
[196]         sleep(1);
[197]     }*/
[198]     pthread_exit(0);
[199] }
[200] void *thread_sensor(void *arg){
[201]     int pin_PIR = 11;
[202]
[203]     // configuração da biblioteca WiringPi
[204]     wiringPiSetup();
[205]
[206]     pinMode(pin_PIR,INPUT);
[207]
[208]     wiringPiISR(0,INT_EDGE_RISING, move_now);
[209]
[210]     while(1) {
[211]         tFim = clock();
[212]         tDecorrido = ((tFim - tInicio) / (CLOCKS_PER_SEC / 1000));
[213]
[214]         //Tempo para que o espelho desligue
[215]         if (tDecorrido > 35000){
[216]             if(b==0){
[217]                 system("pm2 stop mm");
[218]                 b=1;
[219]             }
[220]             a=0;
[221]         }
[222]

```

```
[223]     }
[224]     pthread_exit(NULL);
[225] }
[226] void move_now(void){
[227]     fprintf(stderr,"Movimento\n");
[228]     if(a==0){
[229]         system("pm2 start mm.sh");
[230]         a=1;
[231]         b=0;
[232]     }
[233]     tInicio = clock();
[234] }
```