Despertador Inteligente para Deficientes Auditivos

Projeto Final de Sistemas Embarcados

Arthur Luís Komatsu Aroeira
Engenharia Eletrônica
Universidade de Brasília (FGA)
Gama, DF
email: arthuraroeira@yahoo.com.br

Danovan Martins de Sousa
Engenharia Elerônica
Universidade de Brasília (FGA)
Gama, DF
email: danovanmartins93@gmail.com

***Resumo*—Este relatório apresenta a proposta do projeto final da matéria Sistemas Embarcados e como implementá-lo. Consiste em um despertador inteligente para deficientes auditivos.**
Palavras-chaves — Despertador, sistema embarcado, deficiência auditiva.

I. Introdução

Segundo dados do IBGE de 2013, cerca de 1,1% da população brasileira sofre algum tipo de deficiência auditiva, dentre dos quais 21% têm grau intenso ou muito intenso de limitações, o que compromete atividades habituais [1]. Uma das dificuldades mais notáveis é a dificuldade de acordar em um determinado horário, já que os despertadores convencionais utilizam um sinal sonoro para despertar o usuário.

Além disso, há casos em que o deficiente auditivo reside no mesmo local do que pessoas com problemas de saúde, os quais muitas vezes ficam incapazes de alertá-lo em caso de urgência. Isso ocorreu com Samantha, uma aluna com deficiência auditiva da Universidade de Brasília do curso Engenharia Aeroespacial, como descreve seu depoimento obtido com sua autorização a seguir:

> "Meu nome é Samantha e possuo perda de audição profunda. Estou morando sozinha com minha mãe e recentemente passei por uma situação bem chata. Minha mãe está doente, ela tem vomitado muito e ninguém sabe direito o que é. Ela passou mal de madrugada e eu estava dormindo. Eu acordei de manhã com minha mãe deitada no chão, chorando. Ela tinha se arrastado do banheiro até a porta do meu quarto pra me pedir ajuda. Eu não escuto... Se eu pudesse escutar, teria ajudado minha mãe o quanto antes... Mas só pude ajudar quando minha mãe fez o esforço de sair do banheiro, sem forças, pra me pedir ajuda. Eu queria saber se é possível montar algum sistema de alarme pra mim, no qual minha mãe possa me pedir ajuda. Eu fiquei bem chateada por causa disso, pela minha impotência em ajudar, devido à minha audição."

Diante disso, o relatório apresenta a proposta de uma solução ao problema.

II. Objetivos

Montar e projetar um dispositivo capaz de realizar a função de um despertador para surdos, no qual haverá um alerta de notificações por meio de vibrações ou áudio, sendo que o usuário poderá configurar a intensidade das vibrações para diferenciar o tipo de notificação. O ambiente de configuração contará com um display gráfico, ao qual o usuário poderá configurar o despertador de forma intuitiva.

III. Requisitos

Os seguintes requisitos são esperados para o projeto:

- Notificação para deficientes auditivos;
- Acionamento remoto para emissão de alertas;
- Inserção de alarmes e eventos;
- Sistema de alerta atrvés de vibrações;
- Inclusão do sistema sonoro;

IV. Solução

A seguir, uma breve explicação dos subsistemas eletrônicos do projeto:

A. *Sistema de Interface Gráfica*

O despertador contará com uma interface gráfica para o usuário poder ter a liberdade de:

- Ler o horário atual;
- Alterar data/hora do dispositivo;

- Controlar o sistema de notificações;
- Adicionar/Excluir Alarme;
- Alterar o horário do alarme;
- Alerta de notificação;
- Alerta de emergência;

Escolheu-se uma matriz de LED 7x8 com LEDs azuis de 3mm para o display gráfico.

B. Sistema de Interação com o Usuário

Para o usuário interagir com a interface há uma matriz de LEDS que possui informações do horário, onde através de botões é possível setar as configurações do despertador, tais como horário, data, alarme, etc. O dispostivo será composto de 4 botões, cada um com a respectiva função:

- **Botão 1:** Será possível navegar nas configurações do despertador;
- **Botão 2:** Confirmar ação;
- **Botão 3:** Adicionar parâmetro;
- **Botão 4:** Reduzir parâmetro;

C. Sistema de Notificação ao Usuário

Haverá duas formas de notificação: pelo meio sonoro e vibratório. A pessoa terá liberdade de habilitar e desabilitar cada uma delas. Os dois sistemas estão descritos a seguir:

1) ***Sistema de vibração****:* As vibrações ocorrem por meio de um pequeno motor-dc acoplado a uma carga desbalanceada cuja intensidade pode ser controlada por meio de um sinal PWM. A ideia é inserir o vibrador em uma pequena caixa com fio no qual o usuário poderá inserir confortavelmente embaixo do travesseiro. Assim, ao ser acionado, a cabeça, que é uma parte sensível do corpo, irá vibrar forte o suficiente e, consequentemente, irá despertar a pessoa.

2) ***Sistema de som****:* Aproveitando a saída de áudio do Raspberry Pi, o despertador contará também com um alerta sonoro para caso um usuário sem deficiência auditiva também deseje utilizar seu recurso. Para isso, haverá um pequeno auto-falante acoplado ao amplificador LM386, o qual fornece 125 mW para impedância de 8 Ω [2], suficiente para os requisitos do projeto.

D. Sistema de Acionamento Remoto

O dispositivo possuirá um sistema de alerta inteligente, ao qual pode ser acionado por uma pessoa que esteja no mesmo ambiente. O intuito do projeto é auxiliar pessoas com problemas auditivos, desta forma, se houver um parente próximo que esteja desabilitado de saúde ou que necessite de apoio, sendo um caso de emergência ou alguma atividade urgente, basta pressionar um botão que o despertador irá emitir sinais para que uma notificação seja enviada. Para isso, será utilizado um módulo RF compatível com o Raspberry PI 3 podendo-se adotar um mecanismo para enviar, processar e receber informações. Desta forma, o controle de emissão do sinal contará com um botão e um LED.

E. Controlador Geral

O controlador escolhido para embarcar o sistema foi o Raspberry Pi 3, o qual é um computador que possui um processador ARM de 1.2 GHz 64 bits, 1 Gb de RAM e bluetooth. Ficou popular por ser intuitivo de utilizar, o que fez com que seja utilizado no mundo inteiro em diversos projetos eletrônicos. Possui capacidade de embutir um sistema operacional como o Linux, o que pode tornar o sistema mais estável e com menos memória de uso [3]. Sua escolha se justifica pelo fato de conter todos os requisitos exigidos para cada sistema, tais como:

- Possui entradas e saídas digitais as quais podem ser usadas para controle do sistema de interação com o usuário, sistema de vibração e o sistema de acionamento remoto;
- Possui saída de áudio para o sistema de som;

V. Resultados

Neste tópico, apresenta-se o que foi feito do Ponto de Controle 1 ao Ponto de Controle 4.

A. Hardware

Os Hardwares dos sistemas montados foram:

1) ***Sistema de Interface Gráfica****:* Foi montado uma matriz de LEDs azuis 7x8. A técnica utilizada para controlá-los foi a da multiplexação, na qual ligou-se todos os terminais negativos dos LEDs em uma mesma coluna e todos os terminais positivos na mesma linha, totalizando 7 + 8 = 15 pinos para serem ligados na Raspberry Pi. Dessa forma, é possível acender qualquer LED de forma individual e, consequentemente, qualquer desenho pode ser formado ligando-se cada LED individualmente de forma rápida o suficiente de modo a ser imperceptível ao olho humano.

A seguir, a imagem do protótipo da matriz de LEDs:

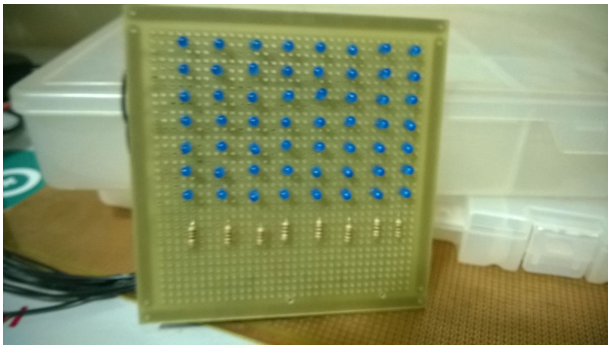

Figura 1. Protótipo da matriz de LED

*2) **Sistema de Vibração**:* Para montar o sistema de Vibração que irá despertar o usuário, foi utilizado os seguintes materiais:

- 3 motores DC 5.9V;
- 1 módulo Rele;
- 1 caixa patola 10cm x 5cm x 3cm;
- 1 cabo de 3 vias.

Os três motores e o módulo relé foram encaixados perfeitamente na caixa. A caixa foi testada por debaixo do travesseiro e não causou nenhum desconforto. Para fazer a caixa vibrar, é necessário acoplar uma carga desbalanceada no motor. Para isso, utilizou-se conectores fêmea-macho que possuem parafusos o que torna possível fixar bem no eixo do motor.

Para acionar os motores, um sinal alto do Raspberry será enviado e, para desligá-los, um sinal baixo será enviado. Como a saída das portas do Raspberry não conseguem fornecer corrente elétrica alta o suficiente, é necessário fazer um driver para seu acionamento. Inicialmente, tentou-se montar uma chave utilizando um transistor bipolar NPN. Porém, os transistores disponíveis não forneceram potência suficiente para acioná-los. Então, por disponibilidade de componentes, foi utilizado um módulo relé, o qual é composto por um relé, porém com alguns incrementos como:

- Driver com transistor para acionar o relé;
- LED indicador de acionamento;
- Diodo para proteção de corrente reversa.

A seguir, o esquemático do circuito montado com o software *Fritzing*:

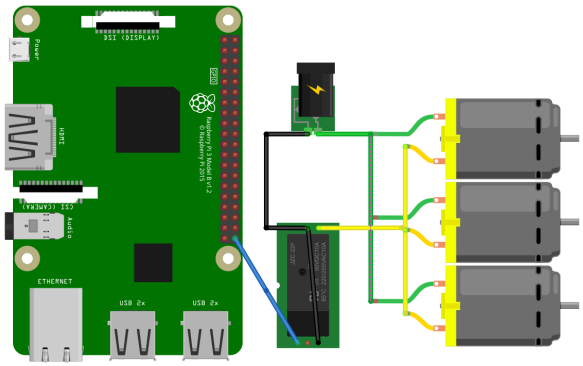

Figura 2. Esquematico do sistema de vibração

A seguir, a foto do sistema final:

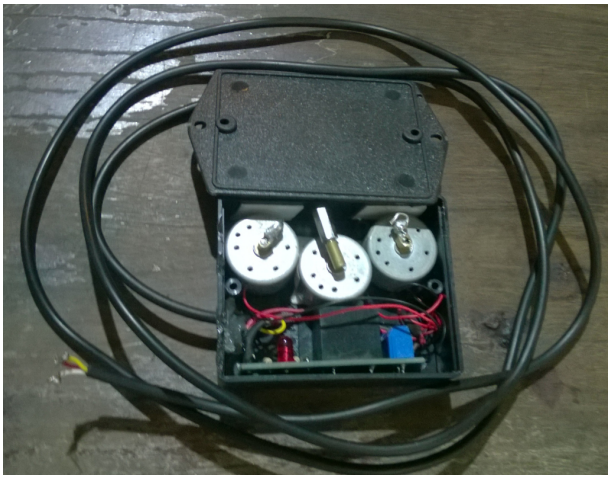

Figura 3. Sistema de vibração

*3) **Sistema de Som**:* O sistema de som foi montado com o CI LM386 com a seguinte placa de circuito impresso obtida em [4]:

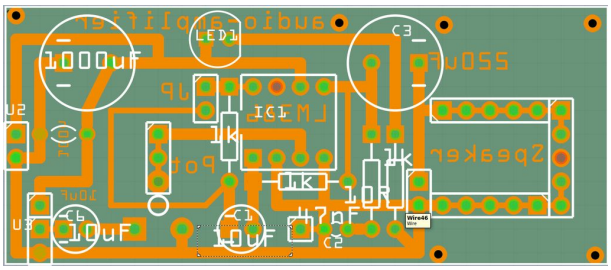

Figura 4. Sistema de vibração

O esquemático do circuito utilizado é:

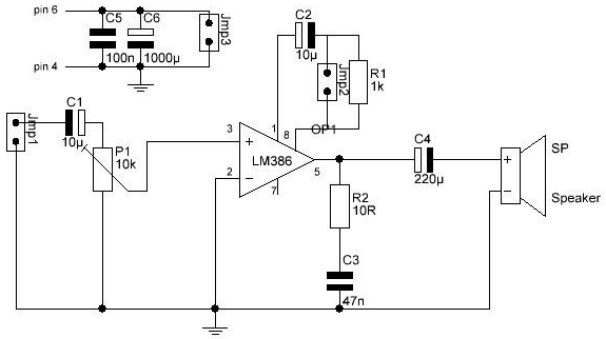

Figura 5. Esquemático do sistema de som

Foram feitas algumas modificações como:

- Foi utilizado um resistor fixo no lugar do potenciômetro, pois o volume será configurado via software;

- Foi retirado o resistor que dá um grave extra;

A alimentação utilizada será de 5V. O sistema funcionou conforme o esperado e o som obtido foi de boa qualidade e com intensidade suficiente para despertar uma pessoa. A seguir, o sistema final montado:

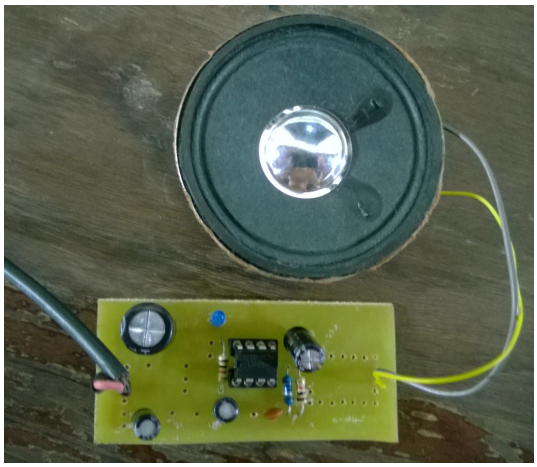

Figura 6. Sistema de som

*4) **Comunicação via Radio Frequência(RF):*** A comunição foi realizada por meio do módulo RF 433Mhz, possuindo um transmissor e um receptor de dados. O transmissor é controlado por um arduino, responsável por enviar um alerta ao usuário quando o botão for pressionado, desta forma o sistema irá acionar o vibrador de forma a alertar o usuário. Foi necessário implementar um conjunto de antenas para cada componente, desta forma foi possível expandir a distância de comunicação entre o transmissor e o receptor.

Os componentes são alimentados com uma tensão de 5 volts, porém o raspberry suporta uma tensão máxima de entrada de 3.3V. Para que o módulo não cause dano ao dispositivo, foi necessário a criação de um divisor de tensão, mantendo a integridade dos dados recebidos.

A comunicação funcionou com sucesso e foi testado a uma distância de 5m com obstáculo. O resultado pode ser aprimorado com o uso de antenas melhores.

B. Software

Para multiplexar a matriz de LEDs, foi necessário a criação de dois processos, os quais estão explicados a seguir:

- **Processo 1:** Consiste em um loop infinito que fica constantemente multiplexando os LEDs de uma variável global "int matriz[7][8]"o qual apenas dois valores são atribuídos: 0 (apagado) e 1 (aceso). Possui atraso de 1 ms entre o acendimento de uma coluna para a próxima. É possível controlar os brilhos dos LEDs criando um sinal de PWM via software, na qual controla-se o delay de quanto um LED fica aceso e apagado, totalizando 1 ms.
- **Processo 2:** Consiste em um loop infinito que injeta uma string alocado numa variável global na matriz. Conforme a matriz vai sendo mostrada, os caracteres vão sendo retirados da variável. É necessário também deslocar a matriz da direita para a esquerda.

Além disso, foi preciso criar um terceiro processo o qual é responsável por executar um comando na shell para executar um sinal sonoro (alarme) quando for acionado. O processo fica em loop infinito e aguarda o valor da variável compartilhada "alarme"= TRUE para emitir o alarme. Quando o comando é executado, ele para o código e espera o áudio ser executado até o final. Como consiste em um processo dedicado a isso, os outros processos podem continuar rodando sem interrupção. Dessa forma, é possível interromper o sinal de alarme quando o usuário apertar o botão de interrupção do despertador.

O primeiro procedimento adotado foi criar uma memória de compartilhamento entre os processos por meio de um arquivo. Para isso, foi utilizada a biblioteca «sys/shm.h>"a qual permite a comunicação entre os processos, utilizando uma estrutura de dados e um endereço compartilhado. As variáveis compartilhadas são:

- **int horas:** Horas do horário atual do relógio,
- **int minutos:** Minutos do horário atual do relógio,
- **int segundos:** Segundos do horário atual do relógio,
- **int desp_hora:** Horas do horário atual do alarme,
- **int desp_minuto:** Segundos do horário atual do alarme,
- **int alarme:** Variável booleana que, quando TRUE, ativa o sistema de vibração e/ou toca um sinal de áudio e FALSE causa o contrário.
- **int ativar_alarme:** Variável booleana que indica se o alarme está ativado.
- **int brilho:** Valor entre 1 e 10 que controla o brilho dos LEDs.
- **int config:** Indica se o despertador se encontra no modo de configuração.
- **int modo:** Variável que escolhe o modo de mostrar o horário no display (decimal, binário e decimal com data).
- **int ativar_musica:** Variável que mostra se o usuário deseja que a música esteja ativada pelo alarme.
- **int parar:** Variável booleana que, quando TRUE, ativa o alarme. Quando FALSE, desativa o alarme.
- **int matriz[7][8]:** Display da matriz de LEDs, contendo apenas 0s e 1s.
- **char *s:** String que será mostrado no display.

Para o setup inicial, foi iniciado o "WiringPiSetup()"para ativar as funções de GPIO da raspberry, habilitando os pinos para o modelo PI-3. Para realizar a comunicação entre processos, a função shmget foi

utilizada, onde é compartilhado um endereço de memória ao qual os processos possuem acesso simutâneo, de forma que possam acessar, sobreescrever e manipular informações sobre as variáveis compartilhadas da struct "memoria_compartilhada", ao qual serão acessadas através da variável x. Após separar um espaço para o compartilhamento de memória, dois processos foram gerados para realizar algumas execuções parciais do programa, desta forma é possível otimizar o desempenho do programa, minimizando erros e falhas.

Para armazenar o horário do sistema foi gerada uma struct "tm *data"que é reponsável por armazenar os valores nas variáveis horas, minutos e segundos. Na mesma foi definido um valor inicial para o despertador, ao qual será acionado às 6:00 horas da manhã caso o usuário não selecione um horário desejado.

As configurações do dispositivos podem ser manipuladas por meio de 4 botões, onde cada botão possui as seguintes funções:

- **B[0]:** Navegar entre as configurações do dispositivo, possibilitando escolher entre: 0-alterar hora; 1-alterar minuto; 2-alterar hora do alarme; 3-alterar minuto do alarme; 4-ativar/desativar alarme; 5-modo de exibição do horário(decimal ou binário); 6-ativar/desativar música; 7-alterar brilho.
- **B[1]:** Confirma a ação realizadas durante o processo de configuração.
- **B[2]:** Incrementa os valores das funções.
- **B[3]:** Decrementa os valores das funções

A biblioteca "../rc-switch/RCSwitch.h"é necessária para realizar a comunicação entre os periféricos do módulo de Rádio Frequência, onde o pino 1 foi setado como entrada de dados para recepção de dados do módulo. O comando "mySwitch = RCSwitch();"é responsável por habilitar as interrupções do módulo RF, fazendo com que o dispositivo esteja recebendo os dados enviados pelo transmissor. O pino de recepção é ativado pela seguinte chamada "mySwitch.enableReceive(PIN);", onde a variável PIN define qual pino do Raspberry irá receber os dados. A tomada de decisão "if (mySwitch.available())"checa se o receptor encontra-se ativo. O receptor envia uma série de pulsos com valores binários que representa um valor numérico, desta forma o alerta só será emitido quando o valor "8591"(escolhido por convenção) for enviado pelo transmissor e armazenado na variável "value", ao qual atualiza os valores por meio do seguinte comando "int value = mySwitch.getReceivedValue();".

O transmissor envia dados apenas quando o botão é acionado. Para minimizar os custos foi utilizado um Arduino, ao qual pode ser substituido por um ATiny ou ATMEGA Stand-Alone. A biblioteca "RCSwitch.h"foi utilizada para realizar a transmissão de dados por meio do arduino. O comando "RCSwitch mySwitch = RCSwitch();"habilita a variável mySwitch para realizar as funções de transmissão do módulo, possibilitando habilitar o pino 10 para transmissão de dados de acordo com o comando "mySwitch.enableTransmit(10);". O botão é estruturado de acordo com a lógica de pull-up, desta forma o uso de resistores externos é desnecessário e os dados são transmitos apenas pino 9, representado pela variável botão, recebe um valor lógico baixo. Para o envio de dados é utilizado o seguinte comando "mySwitch.send(8591, 24);", o primeiro argumento do comando representa o número decimal que será enviado para o receptor, já o segundo argumento define a quantidade de bits em que ocorre o envio da transmissão de dados por meio de uma conversão decimal-binário. Os dados podem ser transmitidos através de bases binárias, decimais ou hexadecimais. Para comunicação serial foi utilizado um baudrate de 9600 bits/s. A seguir, as funções utilizadas no código:

- **main():** O sinal de interrupção por timer SIGALRM é ativado para chamar a função atualizar_horario. Após isso, a main fica em loop infinito e, para evitar processamento desnecessário, é chamado um delay de 1 us.
- **atualizar_horario():** Função responsável por atualizar o valor do horário atual. Chamada a cada 1 segundo pelo sinal de interrupção SIGALRM, a variável segundos é incrementada. Caso o horário atual coincida com o do alarme, o sistema de vibração e o sistema sonoro são ativados. Os sistemas são desativados caso o usuário parte um botão de interrupção ou um minuto se passe.
- **desliga_tudo():** Desliga todos os LEDs da matriz, colocando os terminais negativos no nivel ALTO e os terminais positivos em nivel BAIXO.
- **acende_matriz():** Liga todos os LEDs.
- **pisca_matriz(int n):** pisca todos os LEDs da matriz por 300 ms. É uma forma de informar ao usuário o sucesso do pressionamento de um botão.
- **pisca_numero():** Pisca um número centralizado na tela. Usado para informar o número da configuração ao usuário.
- **apertar_botao(int x):** Retorna TRUE caso o botão x seja pressionado e FALSE caso contrário. Já realiza o delay do DEBOUNCE, assim como espera o usuário soltar o botão para retornar.
- **string Dec_Bin(int n):** Retorna uma string com a representação binária de n.
- **sair():** Sinal SIGINT que sai do programa, fecha a memória compartilhada e desliga os LEDs.

VI. Custos

A tabela I a seguir mostra uma estimativa dos custos necessários para a elaboração do projeto:

Tabela I
TABELA DE CUSTOS

Material	Valor Uni.	Qtde	Total	Fornecedor
Raspberry Pi 3	R$ 200	1	R$ 200	Eletrojun
Motor DC	R$ 2	5	R$ 10	Huinfinito
Módulo RF 400 Mhz	R$ 12,90	1	R$ 12,90	Huinfinito
PCI 20cm x 20cm	R$ 20	1	R$ 20	Huinfinito
Auto falante 0.5 W 8 Ω	R$ 10	1	R$ 10	Huinfinito
Componentes gerais (resistores, capacitores, ...)	-	-	R$ 42	Huinfinito
Total	**R$ 294,90**			

VII. CONCLUSÃO

O protótipo do projeto funcionou com sucesso e não houve nenhum problema no resultado final. É possível juntar todos os periféricos em apenas uma placa de circuito impresso e juntar numa caixa, na qual se teria um produto final totalmente funcional.

Conforme foi listado no depoimento, há uma grande necessidade na implementação de um sistema que emita alertas para pessoas que possuem algum tipo de problema auditivo. Através do dispotivo despertador inteligente para deficientes auditivos é possível emitir alertas que ajuadam a monitorar atividades, desde tarefas básicas até uma situação com caráter emergencial. Desta forma é possível auxíliar pessoas que estejam com a audição parcialmente ou completamente comprometida.

REFERÊNCIAS

[1] VILLELA, Flávia. **IBGE: 6,2% da população têm algum tipo de deficiência**. Agência Brasil, 2015. Disponível em <http://agenciabrasil.ebc.com.br/geral/noticia/2015-08/ibge-62-da-populacao-tem-algum-tipo-de-deficiencia>. Acesso em 02 de abril de 2017.

[2] TEXAS INSTRUMENTS. **LM386 Datasheet**. Disponível em <http://www.ti.com/lit/ds/symlink/lm386.pdf>. Acesso em 03 de abril de 2017. ONE

[3] The MagPi Magazine. **Eben Upton talks Raspberry Pi 3**. Disponível em <https://www.raspberrypi.org/magpi/pi-3-interview/>. Acesso em 03 de abril de 2017.

[4] Instructables. **LM386 Amplifier With PCB**. Disponível em <http://www.instructables.com/id/LM386-amplifier-with-PCB>. Acesso em 06 de maio de 2017.

[5] Wiring Pi. **GPIO Interface library for the Raspberry Pi** Disponível em <http://wiringpi.com/download-and-install/> Acesso em 07 de maio de 2017.

[6] Homautomation **433Mhtz RF communication between Arduino and Raspberry Pi: Raspberry Pi as receiver** Disponível em <http://www.homautomation.org/2013/09/21/433mhtz-rf-communication-between-arduino-and-raspberry-pi/>

ANEXOS

Código Raspberry

```
1
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include "/home/pi/Documents/Embarcados/433Utils/rc-switch/RCSwitch.h"
5  #include <bits/stdc++.h>
6  #include <sys/types.h>
7  #include <string.h>
8  #include <errno.h>
9  #include <unistd.h>
10 #include <wiringPi.h>
11 #include <wiringSerial.h>
12 #include <signal.h>
13 #include <sys/ipc.h>
14 #include <sys/shm.h>
15 #include <termios.h>
16 #include <dirent.h>
17 #include <fcntl.h>
18 #include <bits/stdc++.h>
19 #include <wiringPi.h>
20 #include <signal.h>
21 #include <setjmp.h>
22 #include "caracteres.cpp"
23 using namespace std;
24
25 #define G1 12
26 #define G2 3
27 #define G3 2
28 #define G4 27
29 #define G5 7
30 #define G6 9
31 #define G7 8
32 #define G8 29
33
34 #define V1 25
35 #define V2 24
36 #define V3 23
37 #define V4 22
38 #define V5 21
39 #define V6 14
40 #define V7 13
41
42 #define B1 28 //conf
43 #define B2 26 //Enter
44 #define B3 6 //Baixo
45 #define B4 5 //Cima
46
47 #define PIN 1
48 #define vibrador 4
49
50 #define DESLOCAMENTO 70
51 #define DEBOUNCE 200
52
53 RCSwitch mySwitch;
54
55 typedef struct memoria_compartilhada
56 {
57    int matriz[7][8], horas, minutos, segundos, desp_hora, desp_minuto, alarme, parar,
```

```
      brilho, ativar_alarme, modo, config, parar_escrever, ativar_musica;
 char s[1000];
}dado;

int area;
char str[10];
pid_t pid_filho;
dado *x;
int G[8] = {G1, G2, G3, G4, G5, G6, G7, G8};
int V[7] = {V1, V2, V3, V4, V5, V6, V7};
int B[4] = {B1, B2, B3, B4};

void acende_matriz ();
void desliga_tudo();
void atualizar_horario(int sig);
void multiplexar(int a);
void sair(int a);
void mostrar_horario();
void zera_matriz();
void pisca_matriz();
void pisca_numero(int n);
bool apertar_botao(int x);
bool apertar_botao2(int a);
string Dec_Bin(int n);

int main(int argc, char *argv[])
{
   wiringPiSetup () ;
   pinMode (vibrador, OUTPUT) ;
   for(int i = 0 ; i < 7 ; i++)
      pinMode(V[i], OUTPUT);
   for(int i = 0 ; i < 8 ; i++)
      pinMode(G[i], OUTPUT);
   for(int i = 0 ; i < 4 ; i++)
      pullUpDnControl(B[i], PUD_UP);

   area = shmget(IPC_PRIVATE, sizeof(dado), IPC_CREAT | 0644);
   pid_t pid_filho = fork();
   pid_t pid_filho2 = fork();
   pid_t pid_filho3 = fork();
   x = (dado *) shmat(area, 0, 0);

   struct tm *data;
   time_t tempo;
   tempo = time(NULL);
   data = localtime(&tempo);

   //RF
   int pulseLength = 0;

  // if (argv[1] != NULL)
    // pulseLength = atoi(argv[1]);
   mySwitch = RCSwitch();

   //if (pulseLength != 0)
     // mySwitch.setPulseLength(pulseLength);
   mySwitch.enableReceive(PIN); // Receiver on interrupt 0 => that is pin #2


   //Variaveis iniciais
   x->horas = data -> tm_hour;
   x->minutos = data -> tm_min;
   x->segundos = data -> tm_sec;
   x->desp_hora = 6;
   x->desp_minuto = 0;
```

```
122    x->alarme = 0;
123    x->brilho = 9;
124    x->ativar_alarme = 0;
125    x->modo = 0;
126    x->config = 0;
127    x->parar_escrever = 0;
128    x->ativar_musica = 0;
129
130    if (pid_filho == 0 && (pid_filho2 != 0) && (pid_filho3 != 0))
131    {
132       while(1)
133       {
134
135          if (mySwitch.available())
136          {
137             if(mySwitch.getReceivedValue()==8591)
138             {
139                digitalWrite (vibrador, HIGH);
140                x->parar_escrever = 1;
141                while(!apertar_botao(B[0]) && !apertar_botao(B[1]) && !apertar_botao(B
                       [2]) && !apertar_botao(B[3]) )
142                {
143                   if(x->segundos % 2)
144                      acende_matriz();
145                   else
146                      zera_matriz();
147                   delay(100);
148                }
149             }
150             mySwitch.resetAvailable();
151          }
152
153          if (x->alarme)
154          {
155             x->parar_escrever = 1;
156             digitalWrite (vibrador, HIGH);
157             if(x->ativar_musica)
158             {
159                system("amixer␣sset␣PCM,0␣90%");
160                system("mpg123␣Phenomenon.mp3");
161             }
162          }
163          else
164          {
165             digitalWrite (vibrador, LOW);
166             x->parar_escrever = 0;
167             delay(1);
168          }
169       }
170    }
171    else if(pid_filho2 == 0 && pid_filho != 0 && pid_filho3 != 0)
172    {
173    for(int i = 0 ; 1 ; i = (i + 1) % 8)
174       {
175          for(int j = 0 ; j < 7 ; j++)
176             digitalWrite(V[j], x->matriz[j][i]);
177          digitalWrite(G[i], 0);
178 //   delay(1);
179          delayMicroseconds((x->brilho+1)*100);
180      desliga_tudo();
181          digitalWrite(G[i], 1);
182 //   delay(1);
183          delayMicroseconds(100*(9-x->brilho));
184       }
185    }
```

```
 else if(pid_filho3 == 0 && pid_filho2 != 0 && pid_filho != 0)
 {
  int novo = 0;
     bool stop = 0;
  while(1)
     {
        if(!x->parar_escrever)
        {
      if(strlen(x->s) != 0)
      {
        for(int i =0 ; i < 7 ; i++)
          for(int j = 0 ; j < 7 ; j++)
            x->matriz[j][i] = x->matriz[j][i+1];
        if(novo < 5)
          for(int i = 0 ; i < 7 ; i++)
            x->matriz[i][7] = ASCII[x->s[0]][i][novo];
        else
          for(int i = 0 ; i < 7 ; i++)
            x->matriz[i][7] = 0;
        novo = (novo + 1) % 6;
        if(novo == 0)
          memmove(x->s, x->s+1, strlen(x->s));
              stop = 1;
      }
          else if(stop)
      {
        for(int i = 0 ; i < 7 ; i++)
          x->matriz[i][7] = 0;
        novo = 0;
              stop = 0;
      }
      delay(DESLOCAMENTO);
        }
  }
 }
  else if(pid_filho != 0 && pid_filho2 != 0 && pid_filho3 !=0)
  {
     signal(SIGALRM, atualizar_horario);
// signal(SIGINT, sair);

     alarm(1);
     system("amixer␣cset␣numid=3␣1");
     system("clear");

     desliga_tudo();
     zera_matriz();
  strcpy(x->s, "");
     while(1)
     {
        if(x->alarme)
        {
           while(x->alarme && (!apertar_botao(B[0]) && !apertar_botao(B[1]) && !
               apertar_botao(B[2]) && !apertar_botao(B[3]) ))
           {
              if(x->segundos % 2)
                 acende_matriz();
              else
                 zera_matriz();
           }
           x->ativar_alarme = 0;
        }
        if(x->parar_escrever == 0 && x->alarme == 0 && apertar_botao(B[0]))
        {
           x->config = 1;
           int conf = 0;
```

```
250             pisca_numero(conf);
251             while(1)
252             {
253                if(apertar_botao(B[0]))
254                {
255                   pisca_matriz();
256                   x->config = 0;
257                   break;
258                }
259                if(apertar_botao(B[2]))
260                {
261                   conf == 0 ? conf = 7 : conf--;
262                   pisca_numero(conf);
263                }
264                if(apertar_botao(B[3]))
265                {
266                   conf = (conf + 1) % 8;
267                   pisca_numero(conf);
268                }
269          if(conf < 4)
270          {
271           if(strlen(x->s) == 0 )
272             sprintf(x->s, "␣␣MUDAR␣%s␣␣", conf/2 ? (conf == 3 ? "MINUTO␣DESPERTADOR" : "
                 HORA␣DESPERTADOR") : (conf ? "MINUTO" : "HORA"));
273           if(apertar_botao(B[1]))
274           {
275             strcpy(x->s, "");
276             zera_matriz();
277             while(1)
278             {
279                          x->parar_escrever = 1;
280              for(int i = 0 ; i < 8 ; i++)
281               for(int j = 0 ; j < 7 ; j++)
282               {
283                if(i < 4)
284                 x->matriz[j][i] = ASCII[conf/2 ? (conf == 3 ? x->desp_minuto/10 + '0'
                      : x->desp_hora/10 + '0') : (conf ? x->minutos/10 + '0' : x->horas
                      /10+'0')][j][i+1];
285                else
286                 x->matriz[j][i] = ASCII[conf/2 ? (conf == 3 ? x->desp_minuto%10 + '0'
                      : x->desp_hora%10 + '0') : (conf ? x->minutos%10 + '0' : x->horas
                      %10+'0')][j][i-4];
287               }
288              if(apertar_botao(B[2]))
289               conf/2 ? (conf == 3 ? (x->desp_minuto == 0 ? x->desp_minuto = 59 : x->
                    desp_minuto--) : ( x->desp_hora == 0 ? x->desp_hora = 23 : x->
                    desp_hora--)) : (conf ? (x->minutos == 0 ? x->minutos = 59 : x->
                    minutos--) : ( x->horas == 0 ? x->horas = 23 : x->horas--));
290              if(apertar_botao(B[3]))
291               conf/2 ? (conf == 3 ? x->desp_minuto = (x->desp_minuto + 1) % 60 : x->
                    desp_hora = (x->desp_hora + 1) % 24) : (conf ? x->minutos = (x->
                    minutos + 1) % 60 : x->horas = (x->horas + 1) % 24);
292              if(apertar_botao(B[1]))
293              {
294               zera_matriz();
295               break;
296              }
297              delay(1);
298             }
299                       x->parar_escrever = 0;
300           }
301                }
302          if(conf == 4)
303                {
304                   if(strlen(x->s) == 0)
```

```
305                     strcpy(x->s, "␣␣ATIVAR␣ALARME␣");
306                  if(apertar_botao(B[1]))
307                  {
308                     strcpy(x->s, "");
309                     zera_matriz();
310                     while(1)
311                     {
312                        if(x->ativar_alarme && strlen(x->s) == 0)
313                           strcpy(x->s, "␣␣SIM␣␣");
314                        if(!x->ativar_alarme && strlen(x->s) == 0)
315                           strcpy(x->s, "␣␣NAO␣␣");
316
317                        if(apertar_botao(B[2]) || apertar_botao(B[3]))
318                        {
319                           pisca_matriz();
320                           x->ativar_alarme = !x->ativar_alarme;
321                        }
322                        if(apertar_botao(B[1]))
323                        {
324                           pisca_matriz();
325                           break;
326                        }
327                        delay(1);
328                     }
329                  }
330               }
331         if(conf == 5)
332               {
333                  if(strlen(x->s) == 0)
334                     strcpy(x->s, "␣␣MODO␣");
335                  if(apertar_botao(B[1]))
336                  {
337                     strcpy(x->s, "");
338                     zera_matriz();
339                     while(1)
340                     {
341                        if(x->modo == 0 && strlen(x->s) == 0)
342                           strcpy(x->s, "␣␣DECIMAL␣␣");
343                        if(x->modo == 1 && strlen(x->s) == 0)
344                           strcpy(x->s, "␣␣BINARIO␣␣");
345                        if(x->modo == 2 && strlen(x->s) == 0)
346                           strcpy(x->s, "␣␣DECIMAL␣COM␣DATA␣␣");
347
348                        if(apertar_botao(B[2]))
349                        {
350                           pisca_matriz();
351                           x->modo = (x->modo+1)%3;
352                        }
353                        if( apertar_botao(B[3]))
354                        {
355                           pisca_matriz();
356                           x->modo == 0 ? x->modo = 2 : x->modo--;
357                        }
358                        if(apertar_botao(B[1]))
359                        {
360                           pisca_matriz();
361                           break;
362                        }
363                        delay(1);
364                     }
365                  }
366               }
367         if(conf == 6)
368               {
369                  if(strlen(x->s) == 0)
```

```
                    strcpy(x->s, "␣␣ATIVAR␣MUSICA␣");
                if(apertar_botao(B[1]))
                {
                    strcpy(x->s, "");
                    zera_matriz();
                    while(1)
                    {
                        if(x->ativar_musica == 1 && strlen(x->s) == 0)
                            strcpy(x->s, "␣␣SIM␣␣");
                        if(x->ativar_musica == 0 && strlen(x->s) == 0)
                            strcpy(x->s, "␣␣NAO␣␣");

                        if(apertar_botao(B[2]) || apertar_botao(B[3]))
                        {
                            pisca_matriz();
                            x->ativar_musica = (x->ativar_musica+1)%2;
                        }
                        if(apertar_botao(B[1]))
                        {
                            pisca_matriz();
                            break;
                        }
                        delay(1);
                    }
                }
            }
        if(conf == 7)
            {
                if(strlen(x->s) == 0)
                    strcpy(x->s, "␣␣BRILHO␣␣");
                if(apertar_botao(B[1]))
                {
                    strcpy(x->s, "");
                    zera_matriz();
                    while(1)
                    {
                        if(x->ativar_musica == 0 && strlen(x->s) == 0)
                        {
                            for(int i = 0 ; i < 7 ; i++)
                                for(int j = 1 ; j < 6 ; j++)
                                    x->matriz[i][j] = ASCII[x->brilho+'0'][i][j-1];
                        }
                        if(apertar_botao(B[3]))
                        {
                            x->brilho = (x->brilho+1)%10;
                        }
                        if(apertar_botao(B[2]))
                        {
                            x->brilho == 0 ? x->brilho = 9 : x->brilho--;
                        }
                        if(apertar_botao(B[1]))
                        {
                            pisca_matriz();
                            break;
                        }
                        delay(1);
                    }
                }
            }
          }
        }
      }
    return 0;
  }
}
```

```
435
436 void atualizar_horario(int sig)
437 {
438    x->segundos++;
439    if(x->segundos == 60)
440    {
441       x->segundos = 0;
442       x->minutos++;
443       if(x->minutos == 60)
444       {
445          x->minutos = 0;
446          x->horas++;
447          if(x->horas == 24)
448             x->horas = 0;
449       }
450    }
451
452    printf("%02d:%02d:%02d␣-␣Despertador:␣%02d:%02d:00␣-␣Ativar␣alarme:␣%d␣-␣Ativar␣
        musica:␣%d␣\n", x->horas, x->minutos, x->segundos, x->desp_hora, x->desp_minuto,
         x->ativar_alarme, x->ativar_musica);
453
454    if ((x->horas == x->desp_hora) && (x->minutos == x->desp_minuto) && x->ativar_alarme
       )
455    {
456       x->alarme = 1;
457    }
458    else
459    {
460       x->alarme = 0;
461       system("sudo␣pkill␣mpg123");
462    }
463    if(!x->alarme)
464    switch(x->modo)
465    {
466       case 0:
467         if(strlen(x->s) == 0 && !x->config)
468           sprintf(x->s, "%02d:%02d␣␣", x->horas, x->minutos);
469       break;
470
471       case 1:
472       {
473          string hora = Dec_Bin(x->horas);
474          string minuto = Dec_Bin(x->minutos);
475          string segundo = Dec_Bin(x->segundos);
476          if(!x->config)
477          {
478             zera_matriz();
479             for(int i = 0 ; i < 7 ; i++)
480             {
481                x->matriz[i][0] = hora[i]-'0';
482                x->matriz[i][1] = hora[i]-'0';
483                x->matriz[i][3] = minuto[i]-'0';
484                x->matriz[i][4] = minuto[i]-'0';
485                x->matriz[i][6] = segundo[i]-'0';
486                x->matriz[i][7] = segundo[i]-'0';
487             }
488          }
489       }
490       break;
491
492       case 2:
493          time_t curtime;
494          time(&curtime);
495         if(strlen(x->s) == 0 && !x->config)
496             strcpy(x->s, ctime(&curtime));
```

```
497         break;
498      }
499
500      alarm(1);
501   }
502
503   void pisca_matriz()
504   {
505      x->parar_escrever = 1;
506      strcpy(x->s, "");
507      delay(100);
508      acende_matriz();
509      delay(200);
510      zera_matriz();
511      //delay(300);
512      x->parar_escrever = 0;
513   }
514
515   void pisca_numero(int n)
516   {
517      x->parar_escrever = 1;
518      strcpy(x->s, "");
519      zera_matriz();
520      delay(100);
521      for(int i = 0 ; i < 7 ; i++)
522         for(int j = 1 ; j < 6 ; j++)
523            x->matriz[i][j] = ASCII[n+'0'][i][j-1];
524      delay(200);
525      zera_matriz();
526      //delay(300);
527      x->parar_escrever = 0;
528   }
529
530   void desliga_tudo()
531   {
532      for(int i = 0 ; i < 7 ; i++)
533         digitalWrite(V[i], 0);
534      for(int i = 0 ; i < 8 ; i++)
535         digitalWrite(G[i], 1);
536   }
537
538   void acende_matriz ()
539   {
540      for(int i = 0 ; i < 7 ; i++)
541         for(int j = 0 ; j < 8 ; j++)
542            x->matriz[i][j] = 1;
543   }
544
545   void zera_matriz()
546   {
547      for(int i = 0 ; i < 7 ; i++)
548         for(int j = 0 ; j < 8 ; j++)
549            x->matriz[i][j] = 0;
550   }
551
552   void sair()
553   {
554      zera_matriz();
555      digitalWrite(vibrador, 0);
556      printf("Saindo...\n");
557     shmctl(area, IPC_RMID, 0);
558      exit(1);
559   }
560   bool apertar_botao(int a)
561   {
```

```
  if(!digitalRead(a))
  {
    while(!digitalRead(a));
    delay(DEBOUNCE);
    return 1;
  }
  return 0;
}

string Dec_Bin(int n)
{
   string r;
   for(int i = 0 ; i < 7 ; i++) {r=(n%2==0 ?"0":"1")+r; n/=2;}
   return r;
}
```