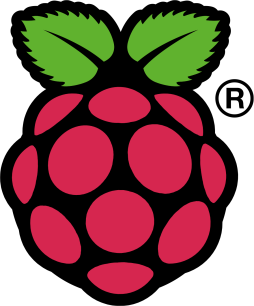
**UTFPR - Universidade Tecnológica Federal do Paraná**

**DAELN – Departamento Acadêmico de Engenharia Eletrônica**

**Professor : Paulo Garcez da Luz**

# Controle de acesso utilizando a Raspberry Pi



Daniel Dalla Vechia

Natalia S. A. Matsushita

Stefam Prestes

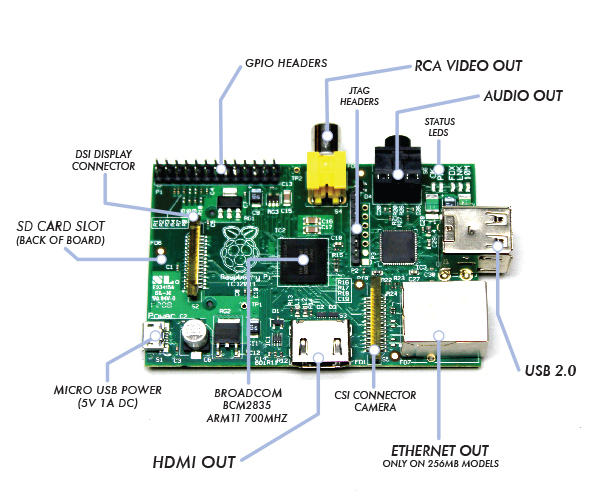
Campo Mourão – PR

2015

1. **Inicialização da Raspberry Pi:**
   1. **Software de simulação RaspBerry Pi para Windows:**

Download do software no link abaixo:

<http://sourceforge.net/projects/rpiqemuwindows/>

**1.2. Instalação:**

**1.2.1. Preparando seu Micro SD:**

Realizar o tutorial disponivel em:

<http://elinux.org/RPi_Easy_SD_Card_Setup>

**1.2.2. Conexões fisicas:**

Realizar a Conexão dos dispositivos a serem utilizados nas seguintes portas:

-Mouse e Teclado: USB 2.0;

-Monitor: HDMI OUT;

-Rede Ethernet: ETHERNET OUT;

-Micro SD: SD CARD SLOT;

-Fonte de Alimentação: MICRO USB POWER;

Após as conexões sua Raspberry estará com as seguintes conexões:

****

**1.3. Configurações básicas:**

Após tudo instalado, iniciando a Raspberry, realizaremos as seguintes configurações:

* Códigos iniciais:

1. Código: apt-get update

Objetivo: para atualizar a data.

1. Código: apt-get install mc

Objetivo: Instalação do mcedit (editor de texto).

1. Código: apt-get upgrade

Objetivo: Atualiza o sistema da Raspberry

Anotações: Processo demorado execute somente se necessário.

* Configura Profile Pi:

1. Código: cd /root/etc

Objetivo: entra no diretório /root/etc.

1. Código: mcedit profile

Objetivo: através do mcedit abre o arquivo profile para edição.

-- No final do arquivo aberto insira:

alias ll='ls -l --color=auto' #insere o atalho do comando ll

alias la='ls -la --color=auto' #insere o atalho do comando la

export EDITOR=mcedit #configura mcedit como editor padrão

export HISTSIZE=1000 #configura o tamanho max. do history

export HISTFILESIZE=1000

--

Anotações: comando ll, lista os arquivos do diretório atual.

* Configurações do root:

1. Código: cd /root

Objetivo: entra no diretório /root

1. Código: mcedit .bashrc

Objetivo: através do mcedit abre o arquivo .bashrc para edição.

-- No final do arquivo insira:

alias ll='ls -l --color=auto' #insere o atalho do comando ll

alias la='ls -la --color=auto' #insere o atalho do comando la

export EDITOR=mcedit #configura mcedit como editor padrão

export HISTSIZE=2000 #configura o tamanho max. do history

export HISTFILESIZE=2000

--

Anotações: comando ll, lista os arquivos do diretório atual.

-> Para confirmar as configurações realizadas é necessário reiniciar o sistema, utilizando o comando abaixo.

1. Código: shutdown -r now

Objetivo: Reinicia o Sistema.

* Configurações Vídeo (tela cheia):

1. Código: cd /boot

Objetivo: entra no diretório /boot

1. Código: mcedit config.txt

Objetivo: abre o arquivo para edição

-- Descomente tirando o caracter '#' antes de:

"disable\_overscan=1"

--

-> Para confirmar as configurações realizadas é necessário reiniciar o sistema, utilizando o comando abaixo.

1. Código: shutdown -r now

Objetivo: Reinicia o Sistema.

* Para retirar a necessidade de login iniciado o raspberry execute:

1. Código: mcedit /etc/inittab

Objetivo: abre o arquivo para edição

-- Dentro deste arquivo insira abaixo de #1:2345:respawn:/sbin... o seguinte comando:

#1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1

--

-> Para confirmar as configurações realizadas é necessário reiniciar o sistema, utilizando o comando abaixo.

1. Código: shutdown -r now

Objetivo: Reinicia o Sistema.

* 1. **Configurações para Programar:**
* Instalar a biblioteca para compilar programas em C.

1. Código: wget www.lt38c.hturbo.com/bcm2835-1.33.tar.gz -o /tmp/bcm2835.tar.gz

Objetivo: Download da biblioteca.

1. Código: tar -zvxf bcm2835-1.33.tar.gz
2. Código: cd bcm2835-1.33
3. Código: ./configure
4. Código: make
5. Código: make check
6. Código: make install
7. Código: cp -R /tmp/bcm2835-1.33/examples/ /root/bcm2835

-- Para compilar os programas dentro da pasta examples utilize:

1. Código: gcc -o blink blink.c -lbcm2835

Objetivo: compila o código blink.c.

* Instalar o servidor mysql:

1. Código: apt-get install mysql-server

Abrirá uma tela azul para a configuração do mysql –server

Ele pedirá uma nova palavra passe para o utilizador “root” do mysql

Digite a nova senha e clique em <OK>

Repita a palavra-passe e clique em <OK>

Aguarde a descompactação e o processamento.

1. ~~Código: sh -c 'echo "<?php phpinfo(); ?>" > /var/www/phpteste.php'~~

~~-- Para testar se a instalação foi bem sucedida, entre no modo gráfico e usando o Midori acesse o phpteste.php~~

* Instalar a bliblioteca wiringPi:

-- Usando do modo gráfico, clique em iniciar > internet > Epiphany e acesse o link: http://wiringpi.com/download-and-install/

-- Execute os passos do “Plano B” sugeridos na página, após baixado mova a pasta para o local desejado e descompacte com:(os números após “wiringPi-XXXXX” provavelmente serão diferentes dependendo da atualização do pacote.

Preste bem atenção na pasta onde você irá salvar o download, você precisará encontrar para executá-la posteriormente.

Retorne a tela de comandos utilizando <Ctrl+Alt+Backspace>

tar xfz wiringPi-f18c8f7.tar.gz

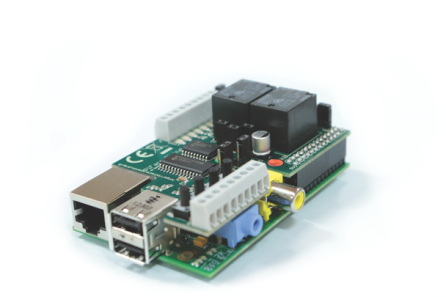
cd wiringPi-f18c8f7

./build

**2. PiFace:**

**2.1.Instalação:**

A PiFace Será conectada fisicamente como demonstra a seguinte Fig.

****

**2.2. Software de simulação:**

O software em que o desenvolvimento foi baseado esta disponível em:

<http://www.piface.org.uk/guides/Install_PiFace_Software/Installing_PiFace_Digital_modules/>

Para instalar siga as instruções disponíveis no link acima, e execute o app que será instalado a partir do modo gráfico.

1. **Instalação da biblioteca BCM2835 e exemplos de softwares em C:**

*Broadcom* BCM2835 consiste em uma biblioteca em C que permite o acesso à *GPIO* e outras funções de *IO* (*Input/Output*) na placa *Rasberry Pi*. Seu uso possibilita o controle e interface com diversos dispositivos externos.

A biblioteca pode ser obtida no site:

<http://www.airspayce.com/mikem/bcm2835/bcm2835-1.38.tar.gz>

Para a instalação, após o *download* da biblioteca (bcm2835-1.xx.tar.gz), seguem os comandos:

**tar zxvf bcm2835-1.xx.tar.gz**

**cd bcm2835-1.xx**

**./configure**

**make**

**sudo make check**

**sudo make install**

* 1. **Funções**

**HIGH** – Seta o pino em nível lógico alto, VERDADEIRO, com 3.3V no pino em questão.

**LOW** – Seta o pino em nível lógico baixo, FALSO, com 0V no pino em questão.

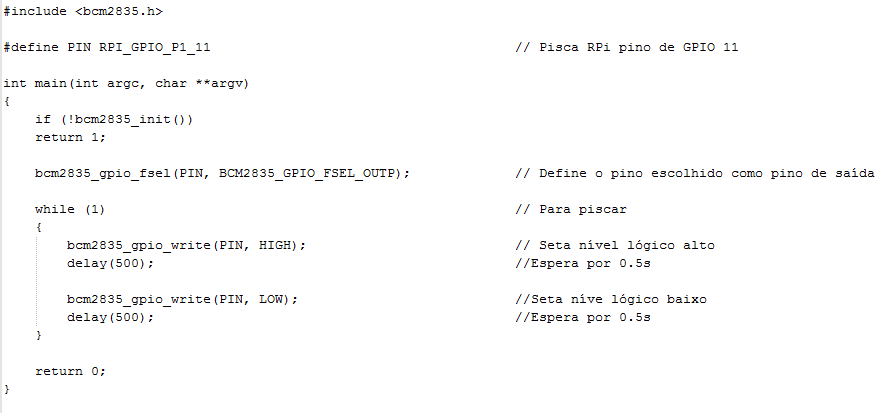
**void bcm2835\_gpio\_fsel(uint8\_t pin, uint8\_t mode)** – Configura o pino em questão como entrada, saída ou outra função.

**void bcm2835\_gpio\_write(uint8\_t pin, uint8\_t on)** – Define o estado do pino em questão.

**1.3 O exemplo Blink.c**

Para desenvolvimento do projeto o exemplo Blink.c foi utilizado como referência no uso dos pinos de GPIO com a biblioteca BCM2835. O exemplo mostra como utilizar da forma correta as funções disponíveis para piscar um LED à cada meio segundo, conectado ao pino 11 da Raspberry Pi. Está disponível, com outros exemplos, em:

<http://www.airspayce.com/mikem/bcm2835/blink_8c-example.html>



Na compilação é necessário utilizar a linha de código:

**gcc –o blink blink.c –l bcm2835**

Para executar:

**sudo ./blink**

1. **Instalação MySQL server:**
2. Código: apt-get install mysql-server

Objetivo: instala o MySQL.

1. Código: mysql -u root –p

Objetivo: Acessa o MySQL.

1. Executar no console: GRANT ALL ON \*.\* TO 'root'@'localhost' IDENTIFIED BY 'password\_mysql\_here';

e

FLUSH PRIVILEGES;

1. Objetivo: Garante acesso do root ao MySQL.

* Para Ativar o acesso ao MySQL em C (rodar uma vez pelo menos):

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get install libmysqlclient-dev
4. Código: mcedit myClient.c
5. Código: g++ myClient.c -o cliente
6. Código: ./cliente
7. **Explicação do código desenvolvido em linguagem de programação C:**
   1. O nosso software trabalha em um “*loop*” infinito no qual incialmente são feitas as verificações de: *update* do *watchdog* pela função db\_update, conexão pela função *db\_open*, identificação do RFID pela função *db\_search\_leituras*, validação do RFID pela função db\_validaRFID e a finalização da conexão pela função db\_close
   2. Ao mesmo tempo um LED de status é programado para piscar a cada 200 milisegundos, com a função de mostrar que o sistema de leitura está corretamente ativado.
   3. Feito o *update* do *watchdog* o software inicia uma verificação do *status* do leitor RFID, sendo que quando o mesmo for igual a “0” (zero) essa leitura é enviada para a “TABELA LEITURAS” a qual é comparada com a “TABELA USUÁRIOS”, tabela onde estão identificados os usuários liberados ou não para o acesso.
   4. Se após essa comparação é identificado que o usuário está habilitado, o software aciona uma porta na qual estará o relé que controla a tranca eletrônica, por um tempo de 10 segundos, acionando também uma segunda porta onde está um LED verde que mostrará que o usuário está com o acesso liberado, por um tempo de 3 segundos.
   5. Caso contrário, ou seja, se o acesso for negado, a porta do relé não é acionada, acionando apenas uma terceira porta, que acenderá um LED vermelho por 3 segundos, identificando ao usuário que seu acesso foi negado.
   6. **TABELA WATCHDOG:**

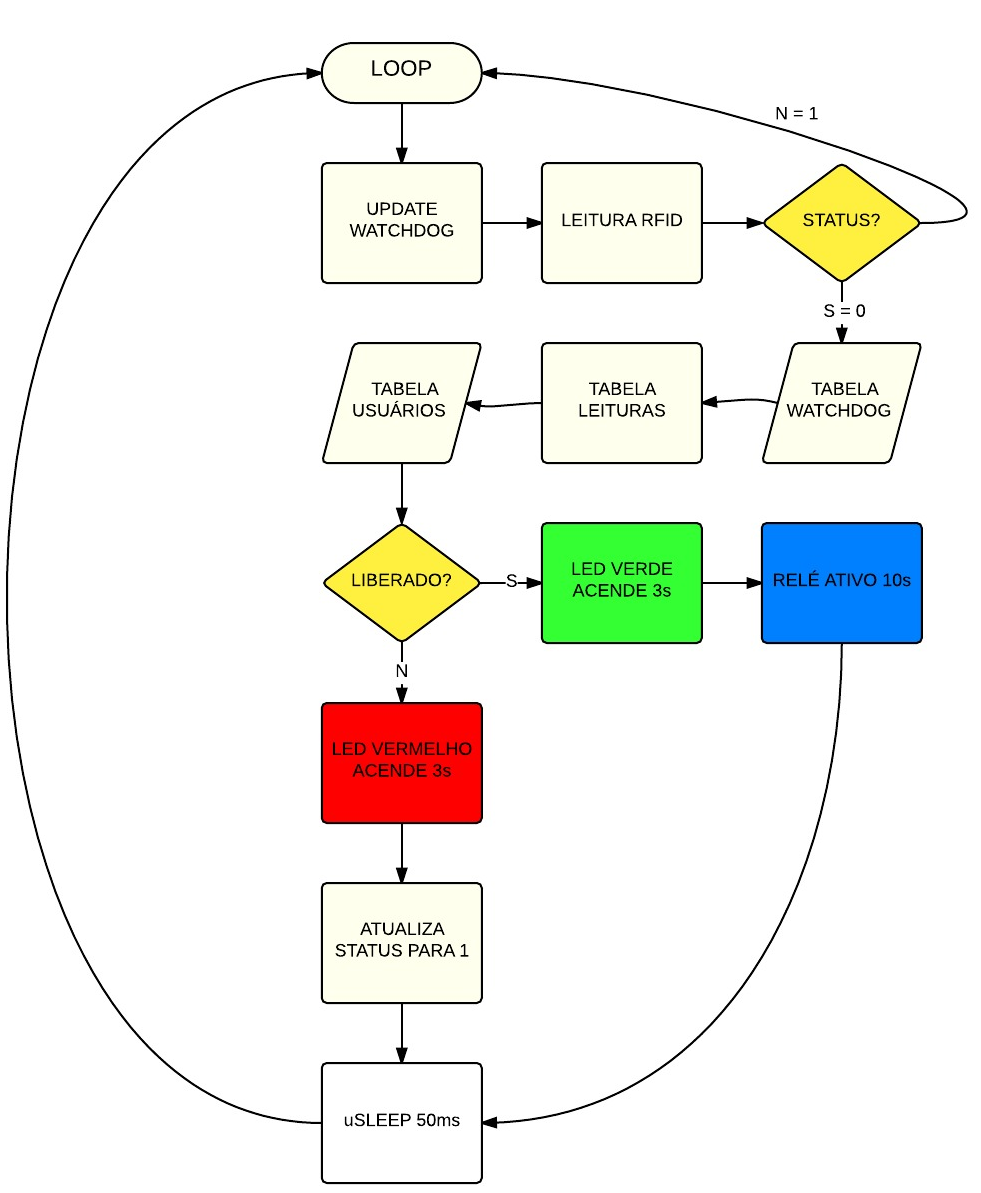
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **HW** | **DATA** | **HORA** |
|  |  |  |  |
|  |  |  |  |

* 1. **TABELA LEITURAS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **HW** | **TAG** | **DATA/HORA** | **STATUS** |
|  |  |  |  | 0 |
|  |  |  |  | 1 |

* 1. **TABELA USUÁRIOS:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **RFID** | **NOME** | **CPF** | **TEL** | **EMAIL** | **GENERO** |
|  |  |  |  |  |  | ALUNO |
|  |  |  |  |  |  | PROF |



1. **Para compilar o código C que foi desenvolvido:**

Comando: gcc xxx.c -lmysqlclient –lbcm2835

Executável: ./a.out

1. **Códigos utilizados:**
2. Código: gcc -o blink blink.c -lbcm2835

Objetivo: compila o código blink.c.

Anotações:

1. Código: history > nomedoarquivo.txt

Objetivo: exportar o histórico

Anotações: o mesmo deve ser enviado por e-mail

1. Código: halt

Objetivo: desliga a raspberry

1. Código: cd [diretório]

Objetivo: muda diretório

Anotações: algumas variações: cd .. (volta um diretório), cd / (vai para diretório raiz).

1. Código: startx

Objetivo: Inicia o modo gráfico.

Anotações: executar somente no usuário PI.

1. Código: ./nomedoexecutavel

Objetivo: executar um programa compilado em C.

1. Código: mcedit nomedoarquivo

Objetivo: abre o editor de texto do arquivo

1. Código: exit

Objetivo: retorna ao usuário PI.

1. Código: clear

Objetivo: limpa os códigos da tela.

1. Código: ps aux

Objetivo: lista os processos em execução

1. Código: kill numerodoprocesso(ex:3046)

Objetivo: encerra o processo descrito

1. Código: find –name nomedoqueprocura

Objetivo: lista todos os arquivos com o nome descrito.

1. Código: ifconfig

Objetivo: mostra o as configurações de rede.

Referências

Sites:

<http://www.raspberrypi.org/documentation/usage/gpio/> - Acessado em 04/11/2014;

<http://wiringpi.com/> - Acessado em 04/11/2014;

<http://www.lt38c.hturbo.com/> - Acessado em 04/11/2014;

**ANEXO I**

Código-fonte implementado:

**#include <stdio.h>**

**#include <string.h>**

**#include <mysql/mysql.h>**

**#include <stdlib.h>**

**#include <sys/time.h>**

**#include <time.h>**

**#include <unistd.h>**

**#include <bcm2835.h>**

**#define PIN10 RPI\_GPIO\_P1\_10 //Pino 10 LED laranja**

**#define PIN07 RPI\_GPIO\_P1\_07 //Pino 07 LED vermelho**

**#define PIN08 RPI\_GPIO\_P1\_08 //Pino 08 LED verde**

**#define PIN18 RPI\_GPIO\_P1\_18 //Pino 18 Rele**

**#define LOCALHOST "www.coele.com.br"**

**#define USUARIO "coelebr\_sistema"**

**#define SENHA "raspberry"**

**#define DATABASE "coelebr\_porta"**

**MYSQL conexao;**

**int horaMilisegundos();**

**int db\_validaRFID();**

**int db\_open();**

**int db\_search\_leituras();**

**int db\_update();**

**int db\_close();**

**int flagNoCard;**

**int tempoA;**

**int tempoB;**

**int validaRFID;**

**int id\_leituras;**

**int idLeituras;**

**int flag1 = 0, flag2 = 0, flag3 = 0, cont1 = 0, cont2 = 0, cont3 = 0;**

**long hrMilli;**

**char query[200] = "SELECT id,tag from leituras where status = '0';";**

**char query2[200];**

**char rfid[100];**

**char buffer[200];**

**int main(int argc, char \*\*argv){**

**if(!bcm2835\_init()){**

**return 1;**

**}**

**else{**

**bcm2835\_gpio\_fsel(PIN10, BCM2835\_GPIO\_FSEL\_OUTP);**

**bcm2835\_gpio\_fsel(PIN07, BCM2835\_GPIO\_FSEL\_OUTP);**

**bcm2835\_gpio\_fsel(PIN08, BCM2835\_GPIO\_FSEL\_OUTP);**

**bcm2835\_gpio\_fsel(PIN18, BCM2835\_GPIO\_FSEL\_OUTP);**

**bcm2835\_gpio\_write(PIN10, LOW);**

**bcm2835\_gpio\_write(PIN07, LOW);**

**bcm2835\_gpio\_write(PIN08, LOW);**

**bcm2835\_gpio\_write(PIN18, LOW);**

**db\_open();**

**for(;;){**

**horaMilisegundos();**

**tempoA = hrMilli;**

**tempoB = tempoA;**

**db\_update();**

**usleep(10);**

**db\_search\_leituras();**

**//strcpy(rfid, "25A9F52D"); //TESTE**

**//flagNoCard = 0;**

**printf("RFID: %s\n", rfid);**

**sprintf(query2, "SELECT \* FROM usuarios WHERE tag = '%s' and horarios ='1'", rfid);**

**db\_validaRFID();**

**horaMilisegundos();**

**tempoA = hrMilli;**

**bcm2835\_gpio\_write(PIN10, HIGH); //LED laranja aceso**

**usleep(10000);**

**bcm2835\_gpio\_write(PIN10, LOW); //LED laranja apagado**

**usleep(50);**

**}**

**}**

**db\_close();**

**}**

**int db\_open(){**

**mysql\_init(&conexao);**

**if(mysql\_real\_connect(&conexao, LOCALHOST, USUARIO, SENHA, DATABASE, 0, NULL, 0))**

**printf("\n\n...\n\n");**

**else{**

**system("echo [`date +%H:%M:%S-%d/%m/%y`] = FALHA DE CONEXAO.\n >> /tmp/completo.log");**

**}**

**}**

**int db\_search\_leituras(){**

**MYSQL\_RES \*resp;**

**MYSQL\_ROW linhas;**

**MYSQL\_FIELD \*campos;**

**int linhasRet;**

**char update\_leituras[200];**

**if (mysql\_query(&conexao, query)){**

**sprintf(buffer, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] = ERRO %s.\n >> /tmp/completo.log", mysql\_error(&conexao));**

**system(buffer);**

**}**

**else{**

**resp = mysql\_store\_result(&conexao);**

**if(resp){**

**linhasRet = mysql\_num\_rows(resp);**

**if(linhasRet > 0){**

**linhas = mysql\_fetch\_row(resp);**

**id\_leituras = atoi(linhas[0]);**

**strcpy(rfid, linhas[1]);**

**//printf("RFID e ID %s %d\n", rfid, id\_leituras); //Teste para verificar o que foi lido na consulta**

**sprintf(update\_leituras, "UPDATE leituras SET status = '1' WHERE id = %d ", id\_leituras);**

**mysql\_query(&conexao, update\_leituras);**

**flagNoCard = 0;**

**}**

**else{**

**flagNoCard = 1;**

**system("echo [`date +%H:%M:%S-%d/%m/%y`] = NO CARD >> /tmp/completo.log");**

**sprintf(rfid, "NULL"); //NULL representa nenhum cartão para consulta**

**}**

**}**

**}**

**}**

**int db\_validaRFID(){**

**//Temporizador 1 - LED verde (3 segundos = 3000 ms).**

**//Temporizador 2 - Rele (10 segundos = 10000 ms).**

**//Temporizador 3 - LED vermelho (3 segundos = 3000 ms).**

**MYSQL\_RES \*resp;**

**MYSQL\_ROW linhas;**

**MYSQL\_FIELD \*campos;**

**int linhasRet;**

**if(flag1 == 1){**

**cont1++;**

**if(cont1 == 5){**

**flag1 = 0;**

**cont1 = 0;**

**bcm2835\_gpio\_write(PIN08, LOW); //LED verde apagado**

**}**

**}**

**if(flag2 == 1){**

**cont2++;**

**if(cont2 == 5){**

**flag2 = 0;**

**cont2 = 0;**

**bcm2835\_gpio\_write(PIN07, LOW); //LED vermelho apagado**

**}**

**}**

**if(flag3 == 1){**

**cont3++;**

**if(cont3 == 16){**

**flag3 = 0;**

**cont3 = 0;**

**bcm2835\_gpio\_write(PIN18, LOW); //Desliga rele**

**}**

**}**

**if (mysql\_query(&conexao, query2)){**

**sprintf(buffer, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =\n\n ERRO %s.\n >> /tmp/completo.log", mysql\_error(&conexao));**

**system(buffer);**

**}**

**else{**

**resp = mysql\_store\_result(&conexao);**

**if(resp){**

**linhasRet = mysql\_num\_rows(resp);**

**if (linhasRet > 0) {**

**system("echo [`date +%H:%M:%S-%d/%m/%y`] = ACESSO LIBERADO! >> /tmp/completo.log");**

**printf("ACESSO LIBERADO\n\n");**

**flag1 = 1;**

**flag3 = 1;**

**bcm2835\_gpio\_write(PIN07, LOW); //LED vermelho aceso**

**bcm2835\_gpio\_write(PIN08, HIGH); //LED verde aceso**

**bcm2835\_gpio\_write(PIN18, HIGH); //Rele acionado**

**}**

**else{**

**if (flagNoCard==0){**

**system("echo [`date +%H:%M:%S-%d/%m/%y`] = BLOQUEADO! >> /tmp/completo.log");**

**printf("BLOQUEADO\n\n");**

**flag2 = 1;**

**bcm2835\_gpio\_write(PIN08, LOW);**

**bcm2835\_gpio\_write(PIN18, LOW);**

**bcm2835\_gpio\_write(PIN07, HIGH); //LED vermelho aceso**

**}**

**}**

**}**

**}**

**}**

**int horaMilisegundos(){**

**struct timeval tv;**

**struct tm \*ptm;**

**long milliseconds;**

**int vtempo[5];**

**gettimeofday (&tv, NULL);**

**ptm = localtime (&tv.tv\_sec);**

**vtempo[0] = ptm->tm\_hour; //Hora**

**vtempo[1] = ptm->tm\_min; //Minuto**

**vtempo[2] = ptm->tm\_sec; //Segundo**

**milliseconds = tv.tv\_usec / 1000; //Milisegundos**

**hrMilli = (vtempo[0] \* 3600000) + (vtempo[1] \* 60000) + (vtempo[2] \* 1000) + milliseconds; //Hora em milisegundos**

**return(hrMilli);**

**}**

**int db\_close(){**

**mysql\_close(&conexao);**

**}**

**int db\_update(){**

**mysql\_query(&conexao, "UPDATE whatdog SET datahora = CURRENT\_TIMESTAMP WHERE pid=2 and hw=1");**

**}**