LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

1.1. A placa Raspberry Pi

Uma visão geral do aspecto da placa Raspberry Pi, bem como algum de seus periféricos podem ser visualizados na Figura 1.1.

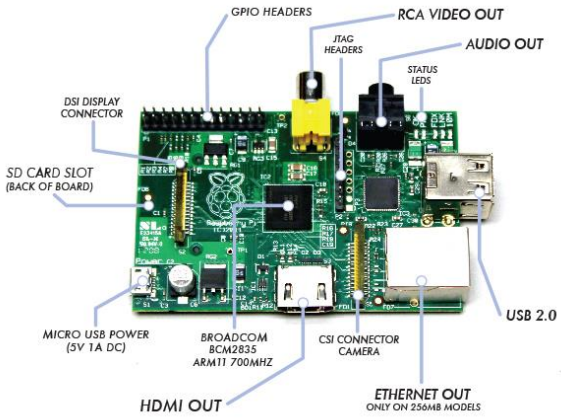

Figura 1.1 – Raspberry Pi

Nos tópicos a seguir iremos apresentar o passo a passo de como montar, configurar e inicializar a sua placa Raspberry Pi afim de deixa-la ao final deste tutorial funcionando perfeitamente com o *Watchdog* ativado.

1.1.1. Preparando o cartão Micro SD

Para este tutorial será utilizado a distribuição Raspbian do Linux que pode ser baixado direto do site da Raspberry. Um tutorial explicando como preparar o cartão SD para utilização na placa pode ser encontrado no link:

http://elinux.org/RPi_Easy_SD_Card_Setup

1.1.2. Conexões Físicas

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:

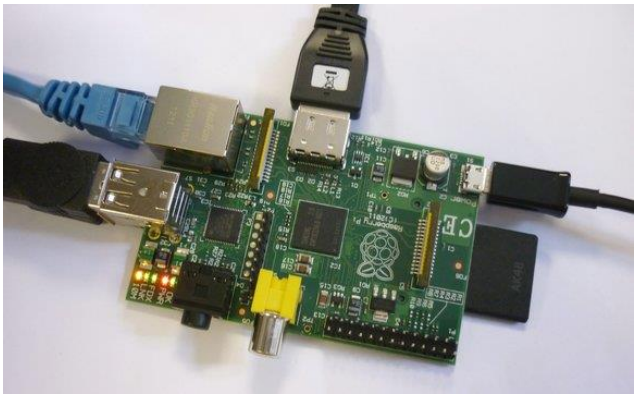

Figura 1.2 – Raspberry com periféricos

1.2. Configurações Iniciais

Assim que for instalado o Raspbian e estiver inicializando pela primeira vez, teremos de realizar algumas configurações inicias que são demonstradas a seguir.

O primeiro passo será atualizar a hora, instalar o *mcedit* (programa que será utilizado para editar arquivos texto) e atualizar o sistema.

```
$ sudo apt-get update
$ sudo apt-get install mc
$ sudo apt-get upgrade
```

Feito isso, a próxima etapa é configurar o profile Pi e Root.

1.2.1. Configurando Profile PI

Abrindo o arquivo profile para as edições:

```
$ cd /etc
$ mcedit profile
```

Ao final do arquivo aberto, adicionar as seguintes linhas:

```
alias ll='ls --color=auto'
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
```

1.2.2. Configurando Profile Root

Abrir o arquivo de configuração:

```
$ sudo su
$ cd /root
$ mcedit .bashrc
```

Ao final do arquivo aberto, adicionar as seguintes linhas:

```
alias ll='ls --color=auto'
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
```

Por fim, para ativar as configurações, reiniciamos o sistema.

```
$ shutdown -r now
```

ou

```
$ reboot
```

1.2.3. Configurando Video

Abrir arquivo de configuração de vídeo:

```
$ cd /boot
$ sudo mcedit config.txt
```

Na linha que estiver escrito “disable_overscan=1”, descomentar tirando o ‘#’ do início da linha.

Ao final, reiniciar sistema.

1.2.4. Login automático

```
$ sudo mcedit /etc/inittab
```

No arquivo “inittab”, comentar a linha que inicia com:

1:2345:respawn:/sbin...

E adicionar na linha subsequente o seguinte comando:

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

Por fim, para ativar as modificações, reiniciar o sistema.

1.3. Instalando biblioteca BCM2835

```
$ wget www.lt38c.hturbo.com/bcm2835-1.33.tar.gz -O
/tmp/bcm2835.tar.gz
$ tar -zvxf bcm2835.tar.gz
$ cd bcm2835-1.33
```

Obs: O nome da pasta pode mudar de acordo com a versão do BCM2835

```
$ ./configure
$ make
$ make check
$ make install
$ cp -R /tmp/bcm2835-1.33/examples/ /root/bcm2835
```

Para testar se a biblioteca foi instalada com exito, entrar na pasta “/root/bcm2835” e compilar o arquivo “blink.c”.

```
$ cd /root/bcm2835
$ sudo gcc -o blink blink.c -lbcm2835
```

1.4. Instalando MySQL Server/client

Para instalar o MySQL server siga os passos abaixo:

- Execute o código abaixo para instalar o MySQL;

```
$ apt-get install mysql-server
```

- Para acessar o MySQL execute o seguinte código;

```
$ mysql -u root -p
```

- No console do MySQL execute os códigos abaixo para garantir o acesso do root ao MySQL;(senha root)

```
$ GRANT ALL ON *.* TO 'root'@'localhost' IDENTIFIED BY
'password_mysql_here';
$ FLUSH PRIVILEGES;
```

- Sequencia de commandos que nao faz sentido, apenas replicado do tutorial do Thiago:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install libmysqlclient-dev
```

1.5. WatchDOG

Entrar na pagina Binerry, Raspberry Pi e segiur os passos:

Incluir e carregar o módulo do watchdog:

```
$ sudo modprobe bcm2708_wdog
$ sudo mcedit /etc/modules
```

Adicionar a linha “bcm2708_wdog” no arquivo modules, em seguida salvar e reiniciar o sistema.

- Baixe os arquivos de teste do watchdog no link:RaspberryPi/snippets
- Execute os códigos abaixo:

```
$ gcc -o wdt_test wdt_test.c
$ sudo ./wdt_test
$ sudo ./wdt_test -t
```

1.5.1. Watchdog “automático”

Caso deseja utilizar o *watchdog* “automático” do sistema, execute os seguintes comandos:

```
$ sudo bash
$ apt-get install watchdog chkconfig
$ chkconfig watchdog on
$ etc/init.d/watchdog start
$ mcedit /etc/watchdog.conf
```

No arquivo watchdog.config inclua a linha:

```
$ watchdog-device = /dev/watchdog
```

1.6. Travando o Linux (Read-Only)

Para garantirmos uma maior vida util do cartão Micro SD, podemos colcoar o sistema Linux no modo ***read-only***. Essa operação além de permitir a maior vida útil, também garante uma maior integridade do sistema, tornando-o robusto contra corrupção de arquivos. O tutorial que foi seguido pode ser encontrado aqui.

2. APÊNDICE

myWatchDog.c

```
/*
============================================================================
 Name        : myWathcDog.c
 Version     : 0.1
 Copyright (C) 2015 by XeXa and Cass, 2015,
 Description :
               A simple watchdog example with mySQL table to control feeding.
============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <linux/watchdog.h>
#include <mysql/mysql.h>

#define SLEEP_TIME 50

// ###############  Parametros para conexao MySQL ########################
#define LOCALHOST "www.coele.com.br"
#define USUARIO "coelebr_sistema"
#define SENHA "raspberry"
#define DATABASE "coelebr_porta"
MYSQL conexao;
char msg[300];
int flagW =0;

// ######################################################################
// ################### PARAMETROS PARA WATHDOG ########################
#define FEED_TIME 10
#define WATCHDOG_TIME 15
#define NUM_PROCESS 10
int deviceHandle;
int disableWatchdog = 1;
int ultimoFeed[NUM_PROCESS];
// ######################################################################



// ############################################################################
// ######################### FUNCOES PARA O MYSQL ###########################
// ############################################################################
//      ### Funcao para ABRIR conexao com o MySQL ###
int db_open (){
    mysql_init(&conexao);
    if(mysql_real_connect(&conexao, LOCALHOST, USUARIO, SENHA, DATABASE, 0, NULL,
0)){
```

```
                // printf("Conectado com sucesso ao DataBase: %s\n", DATABASE);
                sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  Conectado com
sucesso ao DataBase: %s >> /tmp/watchdog.log",DATABASE);
                system(msg);
            }else{
                // printf("Erro ao se conectar com %s no DataBase %s\n",
LOCALHOST,DATABASE);
                sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  Erro ao se
conectar com %s no DataBase %s >> /tmp/watchdog.log",LOCALHOST,DATABASE);
                system(msg);
            }
        }

        //      ### Funcao para FECHAR conexao com o MySQL ###
        int db_close(){
            mysql_close(&conexao);
        }

        int db_timeDiffAll (int tempo){
            char query[200];
            MYSQL_RES *resp;
            MYSQL_ROW linha;

        //    db_open();
            mysql_query(&conexao,"UPDATE whatdog SET datahora = CURRENT_TIMESTAMP WHERE
pid=3 and hw=1");
            sprintf(query, "SELECT TIME_TO_SEC(TIMEDIFF(CURRENT_TIMESTAMP,datahora))>%d
as UltimaRespsota FROM whatdog;",tempo);
            if (mysql_query(&conexao, query)){
                // printf("\nErro: %s\n",mysql_error(&conexao));
                sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  Erro: %s >>
/tmp/watchdog.log",mysql_error(&conexao));
                system(msg);
            }else{
                resp = mysql_store_result(&conexao);
                if (resp){
                    int aux = 0;
                    while (linha = mysql_fetch_row(resp)){
                        ultimoFeed[aux] = atoi(linha[0]);
                        aux++;
                    }
                }
            }

        //    mysql_query(&conexao, query);
        //    db_close();
        }
        // ############################################################################
```

```
        // ###############################################################################
        // ###############  INICIALIZACAO DO PARAMETROS PARA O WATCHDOG  ##############$###
        // ###############################################################################
        int init_WDog (int argc, char *argv[], int tempo){

            // test watchdog reset via t-param
            if (argc > 1) {
               if (!strncasecmp(argv[1], "-t", 2)) {
                   disableWatchdog = 0;
               }
            }

            // printf("Disabling watchdog before closing device: %d\n", disableWatchdog);
            sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  Disabling watchdog
before closing device: %d >> /tmp/watchdog.log",disableWatchdog);
            system(msg);

            // open watchdog device on /dev/watchdog
            if ((deviceHandle = open("/dev/watchdog", O_RDWR | O_NOCTTY)) < 0) {
               // printf("Error: Couldn't open watchdog device! %d\n", deviceHandle);
               sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  Error: Couldn't
open watchdog device! %d >> /tmp/watchdog.log",deviceHandle);
               system(msg);
               return 1;
            }

        //      ################ CONFIGURANDO TIMER DO WATCHDOG ###################
            int timeout = tempo;                                            // TEMPO EM
SEGUNDOS
            ioctl(deviceHandle, WDIOC_SETTIMEOUT, &timeout);
            ioctl(deviceHandle, WDIOC_GETTIMEOUT, &timeout);
            // printf("The watchdog timeout is %d seconds.\n\n", timeout);
            sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  The watchdog timeout
is %d seconds. >> /tmp/watchdog.log",timeout);
            system(msg);
        }


        int init(int argc, char *argv[]){
            // init UltimoFeed
            int z;
            for (z=0; z<NUM_PROCESS; z++){
                ultimoFeed[z] = 0;
            }
            init_WDog(argc,argv,WATCHDOG_TIME);
        }
        // ###############################################################################
```

```
        // ################################################################################
        // ################  FUNCAO DE PROCESSAMENTO DO WATCHDOG  ########################
        // ################################################################################
        int feedDog(){
            int z=0;
            int flag = 0;

            while (z<NUM_PROCESS && flag==0){
                flagW++;
                if(ultimoFeed[z]==1){
                    flag=1;
                    // printf("\n\n\n\n\n\t\t MORREU = %d\t\t \n\n\n\n\n",z);
                   sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  MORREU = %d
>> /tmp/watchdog.log",z);
                    system(msg);
                }
                z++;
            }
            if (flag==0){
               ioctl(deviceHandle, WDIOC_KEEPALIVE, 0);
                // printf("\n\n\n\n\n  TUM S2 s2 S2 s2  TUM S2 s2 S2 s2.\n\n\n\n\n\n");

                if (flagW>100){
                    sprintf(msg, "echo [`date +%%H:%%M:%%S-%%d/%%m/%%y`] =  TUM S2 s2 S2
s2  TUM S2 s2 S2 s2 >> /tmp/watchdog.log");
                    system(msg);
                    flagW=0;
                }
            }
        }
        // ################################################################################
        // ################################################################################

        void main (int argc, char *argv[]){
            db_open();
            init(argc,argv);
            while(1){
                db_timeDiffAll(FEED_TIME);
                feedDog();
                usleep(SLEEP_TIME);
            }
            db_close();
        }
```