UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CAMPO MOURÃO

GERÊNCIA DE PESQUISA E GRADUAÇÃO

DISCIPLINA DE SISTEMAS EMBARCADOS – LT38C
WWW.LT38C.HTURBO.COM

FABIO FUDOLI, LOUISIE STANISZEWSKI, LUCAS BLESSA

COMUNICAÇÃO COM ATMEGA328 VIA TRANSMISSÃO UART

PROJETO FINAL DE DISCIPLINA

CAMPO MOURÃO, DEZEMBRO 2015

FABIO FUDOLI, LOUISIE STANISZEWSKI, LUCAS BLESSA

COMUNICAÇÃO COM ATMEGA328 VIA TRANSMISSÃO UART

Trabalho de conclusão de disciplina apresentada ao Professor de Sistemas Embarcos no curso de Graduação em Engenharia Elétrônica da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção da aprovação na disciplina.

Orientador: Prof. Msc. Paulo Denis Garcez da Luz

CAMPO MOURÃO, DEZEMBRO 2015

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ARM - Advanced RISC Machine
SoC - System on a Chip
UART - Universal Asynchronous Receiver/Transmitter
CRC - Cyclic Redundancy Check Byte

RESUMO

O presente trabalho apresenta o desenvolvimento de um sistema capaz de efetuar a conexão da placa RaspiberryPI a seu shield intitulado de GertBoard.

CAPÍTULO 1
INTRODUÇÃO

1.1. SISTEMAS EMBARCADOS

Um sistema embarcado é um sistema microprocessado, onde o computador é completamente integrado ou dedicado somente a um dispositivo, ou sistema que ele controla. Este sistema é considerado independente e capaz de realizar um determinado objetivo em sua estrutura funcional.

Diferentemente dos computadores pessoais, um sistema embarcado realiza um conjunto de tarefa previamente definidas. E com isso, pode-se otimizar o projeto reduzindo seu tamanho, recursos computacionais e custo do produto. De maneira geral tornar um sistema embarcado é torna-lo mais flexível, mais funcional devido a sua alta gama de aplicações e inovações.

Como o próprio nome prediz embarca inúmeros outros sistemas em uma única central de processamento, como exemplo o controle de um subsistema de uma planta industrial, monitorando via hardware (sensores, teclados, motores), gerenciado por um único sistema. Quem geralmente gerencia toda a desenvoltura dos processos é um sistema operacional, tal como em um computador pessoal.

Por praticidade a placa de desenvolvimento *Raspberry Pi* utiliza uma versão de código aberto, baseado no kernel do Linux.

1.2. RASPBERRY PI

A Raspberry Pi é um computador do tamanho de um cartão de crédito, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão, todo o hardware é integrado numa única placa. Pode ser considerado um computador portátil, foi desenvolvido no reino unido pela fundação Raspberry Pi® e lançada no início de 2012.

É um pequeno dispositivo capaz de fazer tudo que você esperaria de um computador desktop, como navegar na internet, reproduzir vídeo de alta definição, fazer planilhas, processamento de texto, e jogar jogos.

A Raspberry Pi é uma plataforma de desenvolvimento capaz de realizar inúmeras tarefas e utilidades, é uma máquina completa com um núcleo de arquitetura ARM (Advanced RISC Machine) capaz de processar diversos processos. Igualmente a uma placa mãe é uma placa de circuito impresso que possui um microprocessador que é gerido por um sistema operacional. O núcleo é um processador multimídia Broadcom BCM2835 tecnologia SoC

(System on a Chip), isso significa que toda a unidade de processamento está contida em um único chip, oculto por uma memória de 512 MB, localizado no centro da placa.

1.3. MOTIVAÇÕES

Dado a crescente busca por tecnologias cada vez mais acessíveis e por concorrências mais acirradas, a utilização de sistemas embarcados tem crescido muito, devido a sua alta eficiência e custo relativamente baixo. Dessa forma sendo necessário o estudo e desenvolvimento de projetos capazes de gerir, executar e monitorar inúmeras grandezas nas mais diversas escalas e setores. Tais projetos podem ser possíveis, principalmente no setor industrial e educacional, utilizando de plataformas de desenvolvimento, como a *Raspberry Pi*, uma aplicação de sistemas embarcados.

1.4. OBJETIVOS

O objetivo desse trabalho trata-se de efetuar a conexão da placa RaspiberryPI a seu shield intitulado de GertBoard. Através da GertBoard adiciona-se funcionalidades a RaspiberryPI. Com destaque para um microcontrolador da ATMEL modelo Atmega328p carregado com o bootloader da plataforma Arduino que está presente na GertBoard.

Com isso, é possível efetuar uma comunicação com microcontrolador presente na GertBoard através de comunicação serial UART com a RaspiberryPI. Essa comunicação é baseada em um protocolo que será descrito nesse trabalho. Através da conexão UART, será possível acionar leds desejados que estão presente na GertBoard.

A comunicação da RaspiberryPI com a GertBoard exemplifica seu funcionamento como mestre em um sistema de comunicação onde deseja-se controlar periféricos conectados à RaspiberryPI da maneira desejada através de comunicação UART utilizando um protocolo próprio para troca de dados entre ambos.

1.5. ESTRUTURA DO TRABALHO

O trabalho será dividido em 4 capítulos, no primeiro será feita uma breve explicação sobre sistemas embarcados e a plataforma de desenvolvimento, juntamente com objetivos e motivações.

Já no segundo capitulo estarão presentes as instalações necessárias e os primeiros ajustes necessários, juntamente com as principais bibliotecas.

O terceiro capitulo irá conter a descrição do programa principal adaptado, para a realização do projeto e sua funcionalidade.

E o por último um quarto capitulo para explicar sucintamente a funcionalidade da Gertboard e como fazer a instalação na *Raspberry*.

CAPÍTULO 2

INSTALAÇÂO DO SISTEMA INICIAL

2.1. INTRODUÇÃO

Sistemas operacionais são softwares que possuem como objetivo otimizar o funcionamento de um computador, dispositivo que é constituído por vários outros dispositivos que necessitam de informações especificas para o correto funcionamento. Os mais populares sistemas operacionais são: Windows, Linux e Mac OS X.

O sistema Linux que será utilizado é o Debian, pois é um sistema operado na plataforma GNU/Linux que é um sistema operacional de código aberto, o sistema operacional em questão é o Raspbian que é uma variação do Debian para operar na Raspberry, e será utilizado por ele ser um software livre, e oferecer mais de 35.000 pacotes deb de software, que estão pré-compilados, para serem facilmente instalados na Raspberry Pi.

2.2. DESCRIÇÃO SUSCINTA DAS INTALAÇÔES

2.2.1. INSTALAÇÃO DO SISTEMA OPERACIONAL

Primeiramente inicia-se a instalação no Micro-SD, de pelo menos 4gb, realizando o download do sistema operacional Raspbian disponível para download no seguinte endereço web:

www.raspberrypi.org/downloads/

Ao se acessar o site pode-se realizar o download, apresentado na Figura 1.

Figura 1 - Interface de download Raspbian

Logo deve ser feito o download do software Win32DiskImager, o qual realizará a gravação do sistema operacional no Micro-SD, a tela para download pode ser vista na Figura 2 e o mesmo pode ser encontrado no seguinte endereço web:

sourceforge.net/projects/win32diskimager/

Figura 2 - Campo de Download Disk Imager

2.2.2. LIGANDO A RASPBERRY PI

Após o sistema operacional instalado, já se pode ligar a placa *Raspberry PI*. A placa oferece a conexão de vários periféricos, como visto na figura 3.

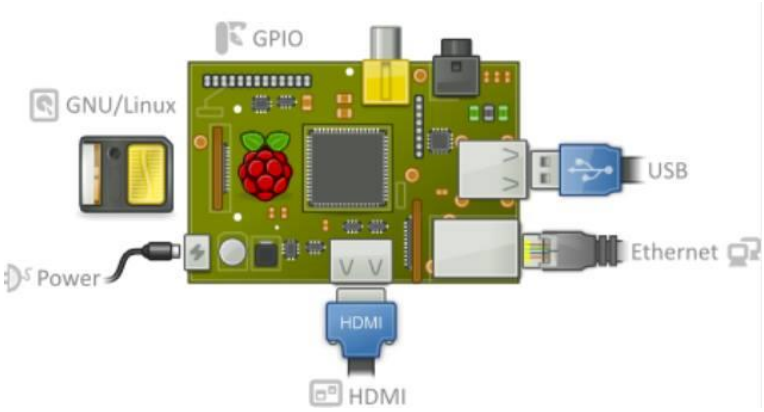

Figura 3 - Layout Raspberry Pi

2.2.3. CONFIGURAÇÕES INICIAIS DO SISTEMA

Após a instalação e a conexão dos periféricos necessários, ao inicializar pela primeira vez o sistema operacional aparecerá a tela de configurações da *Raspberry PI*, denominada Rasp-Config como visto na figura 4.

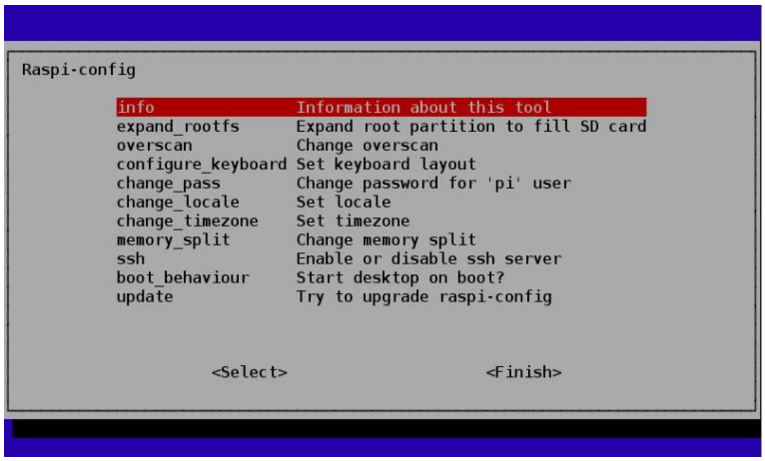

Figura 4- Tela de configuração Raspberry Pi

Logo, pode-se iniciar a configuração da expansão da partição raiz do cartão SD, com o objetivo de tornar toda a memória utilizado, confirmando com a opção em “OK”.

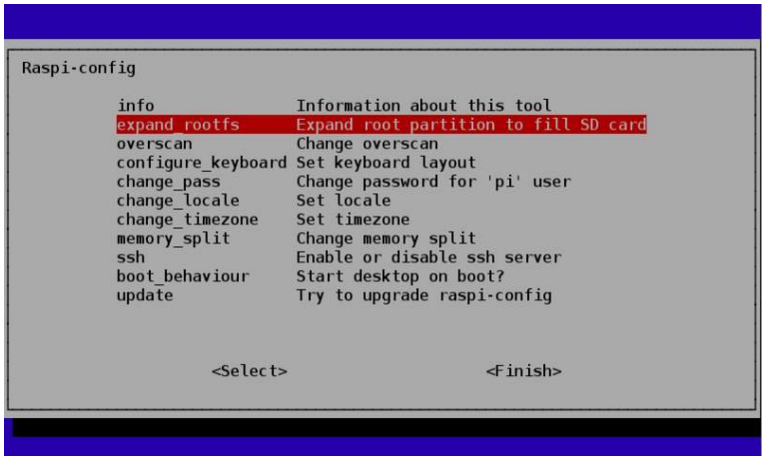

Figura 5 - Primeira Configuração

Figura 6 - Confirmação de configuração.

Em primeiro momento o software não irá utilizar toda a tela do monitor, para trabalhar com a tela cheia siga os seguintes passos:

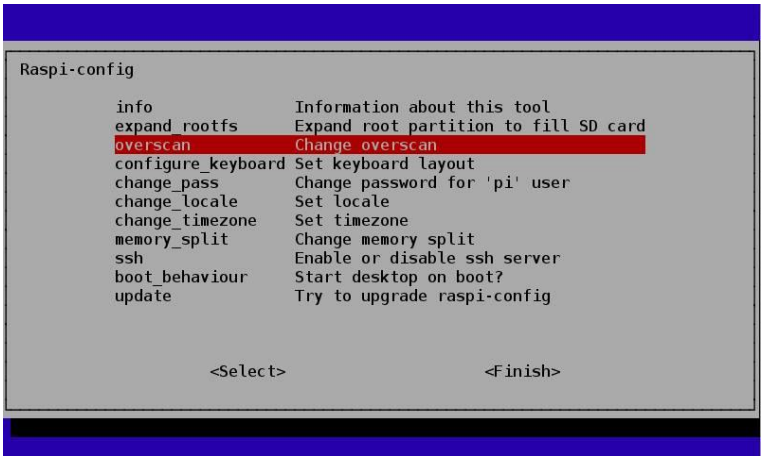

Figura 7 - Overscan.

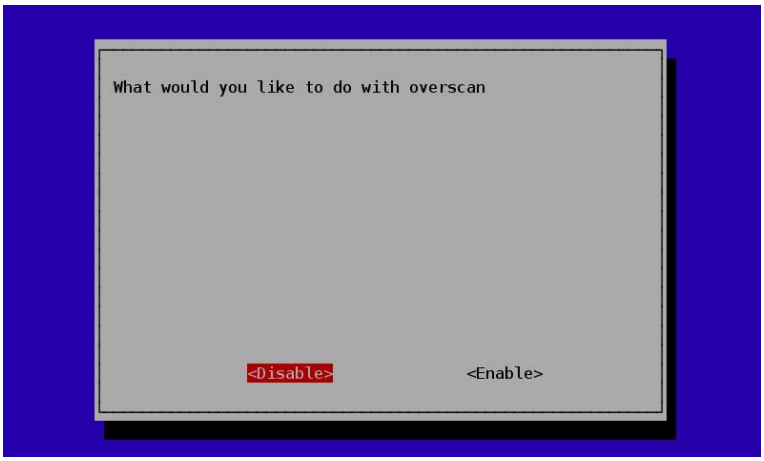

Figura 8 - Overscan.

O passo seguinte será a configuração do teclado, nesse caso foi utilizada a configuração do teclado padronizado brasileiro. Segue o passo a passo da figura 3 até a figura 7.

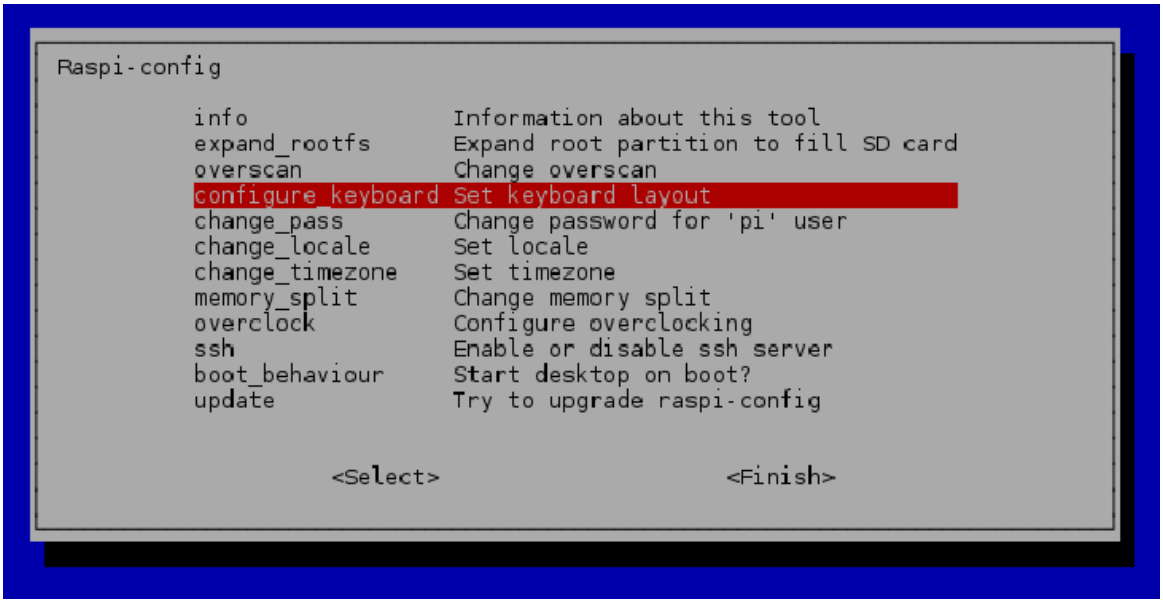

Figura 9 – Configuração teclado

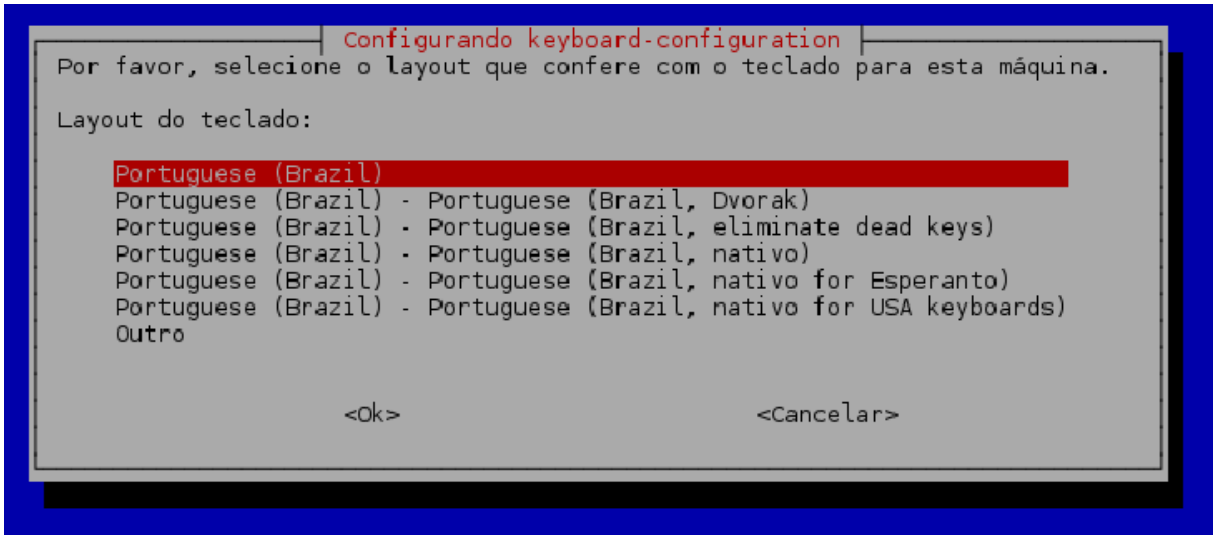

Figura 10 - Configuração do teclado

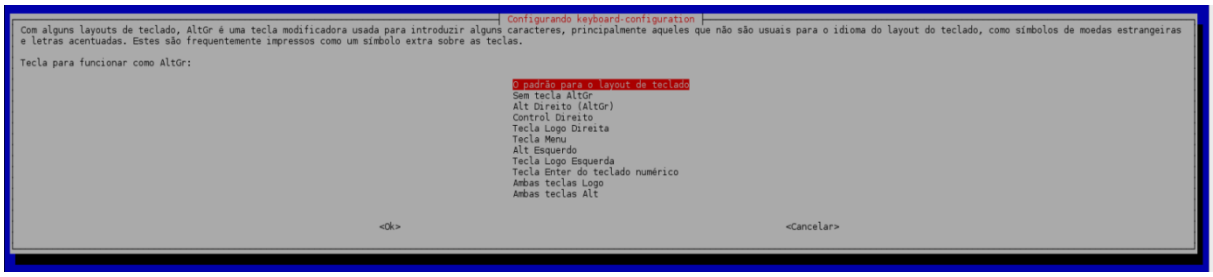

Figura 11– Configuração do teclado

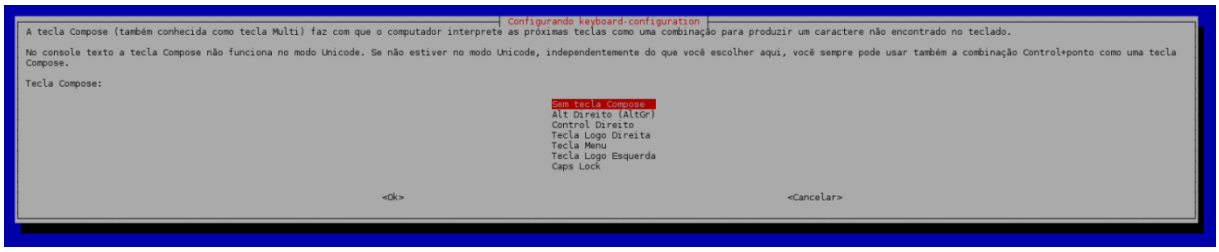

Figura 12– Configuração do teclado

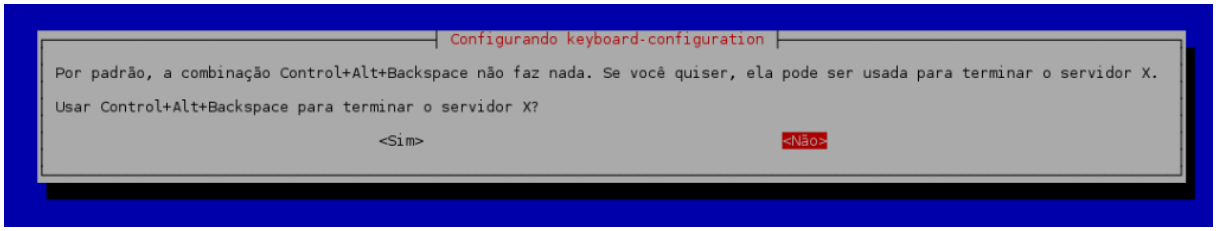

Figura 13– Configuração do teclado

A próxima etapa será definir o país em que se encontra a raspberry para se definir o idioma da mesma. Para esta etapa segue-se as figuras de 8 a 10

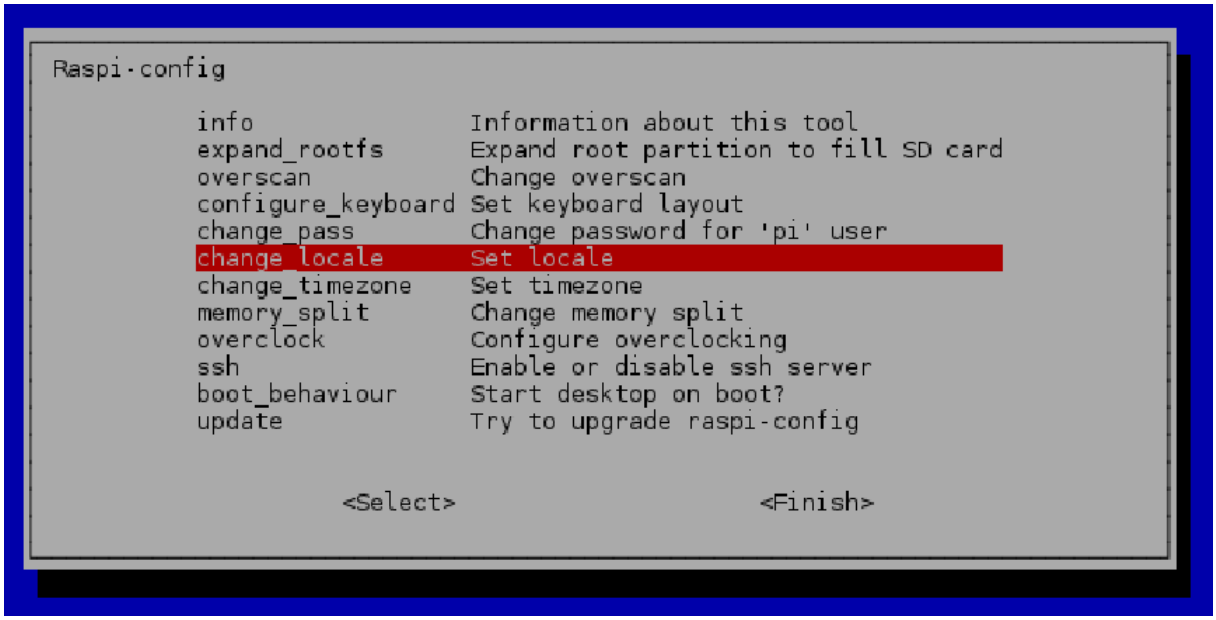

Figura 14– Configuração da localização

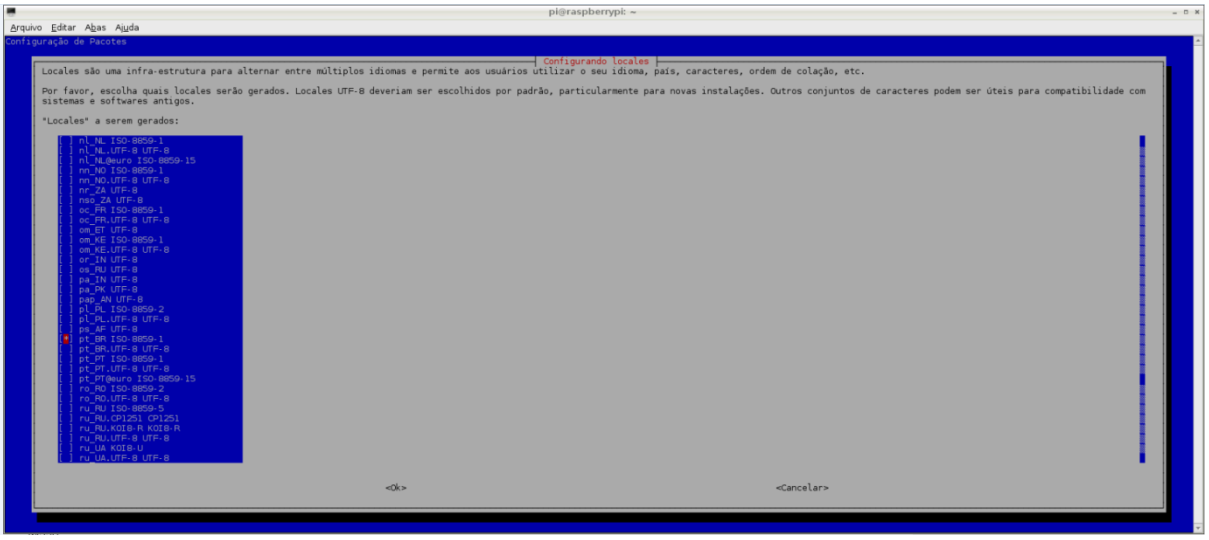

Figura 15– Configuração da localização

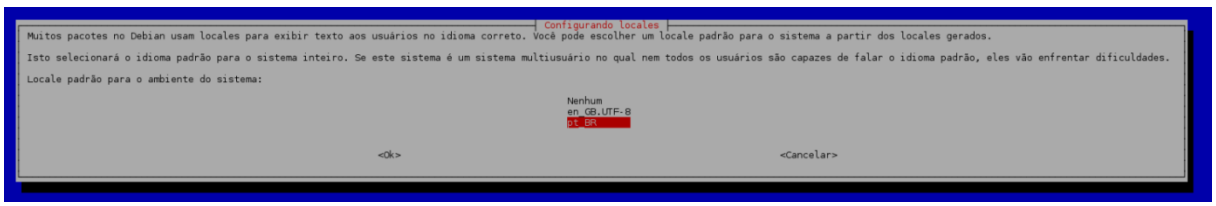

Figura 16– Configuração da localização

A próxima etapa de selecionar o fuso horário da raspberry, pois ela utiliza isso para atualizar sua data e hora, pois ela os atualiza via internet, uma vez que a mesma não possui bios.

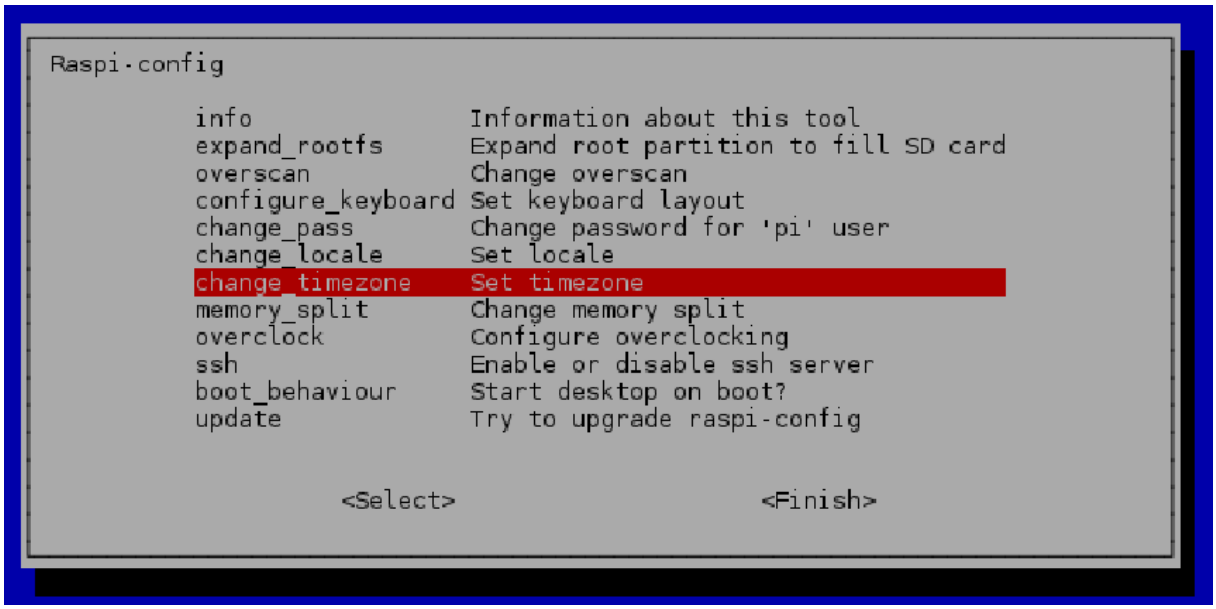

Figura 17– Configuração da timezone

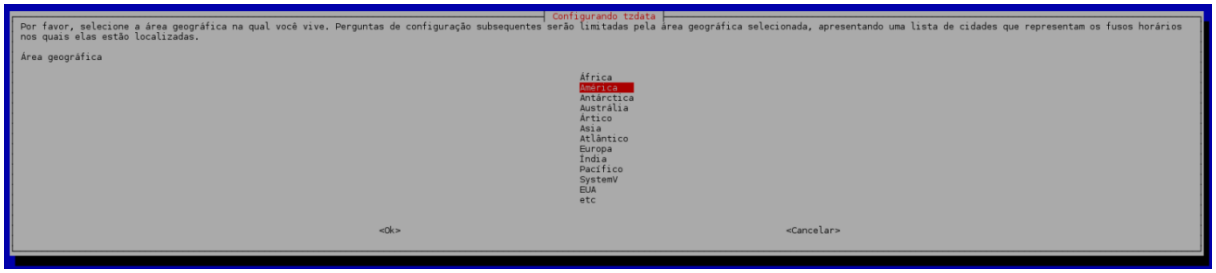

Figura 18– Configuração da timezone

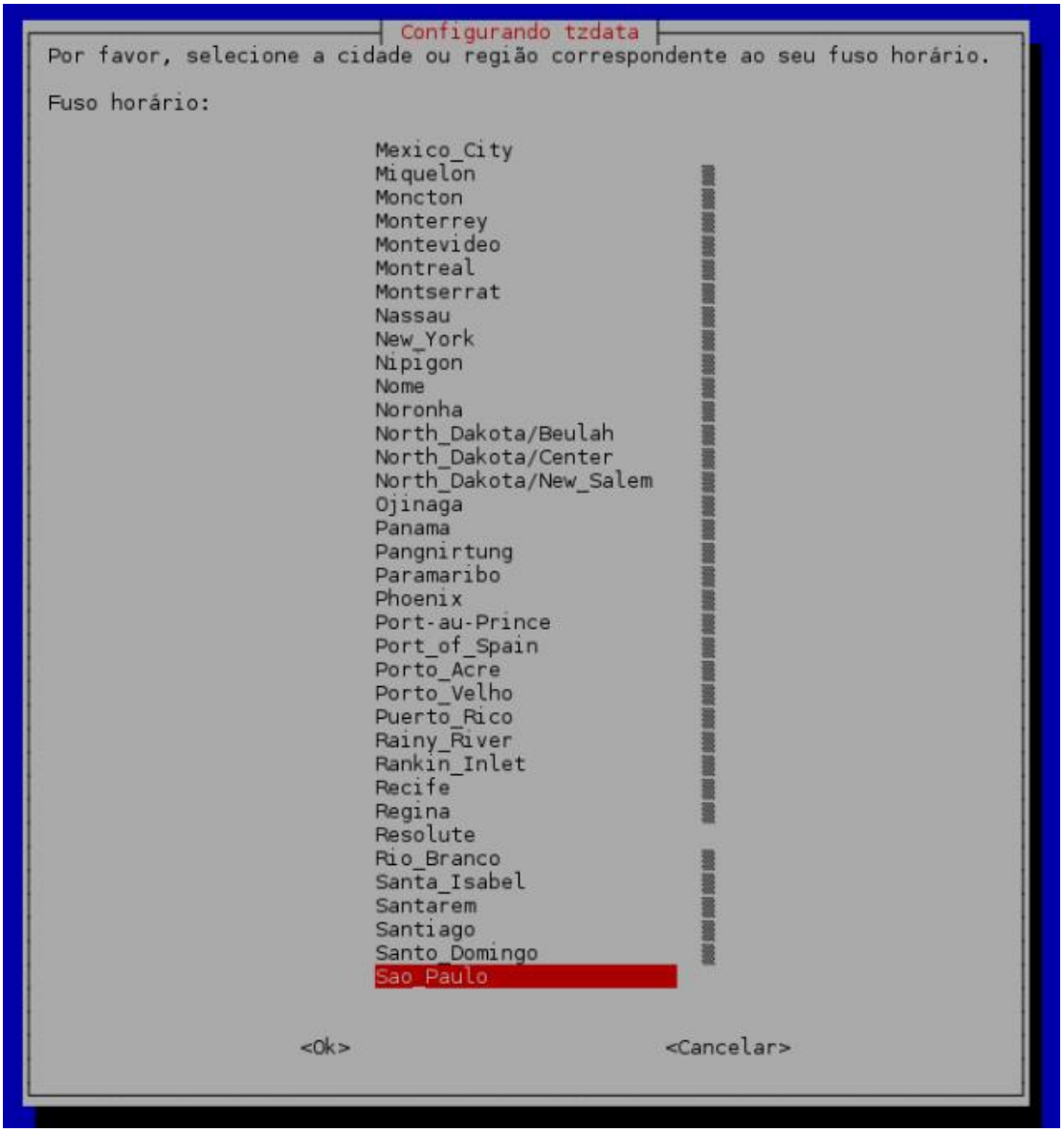

Figura 19– Configuração da timezone

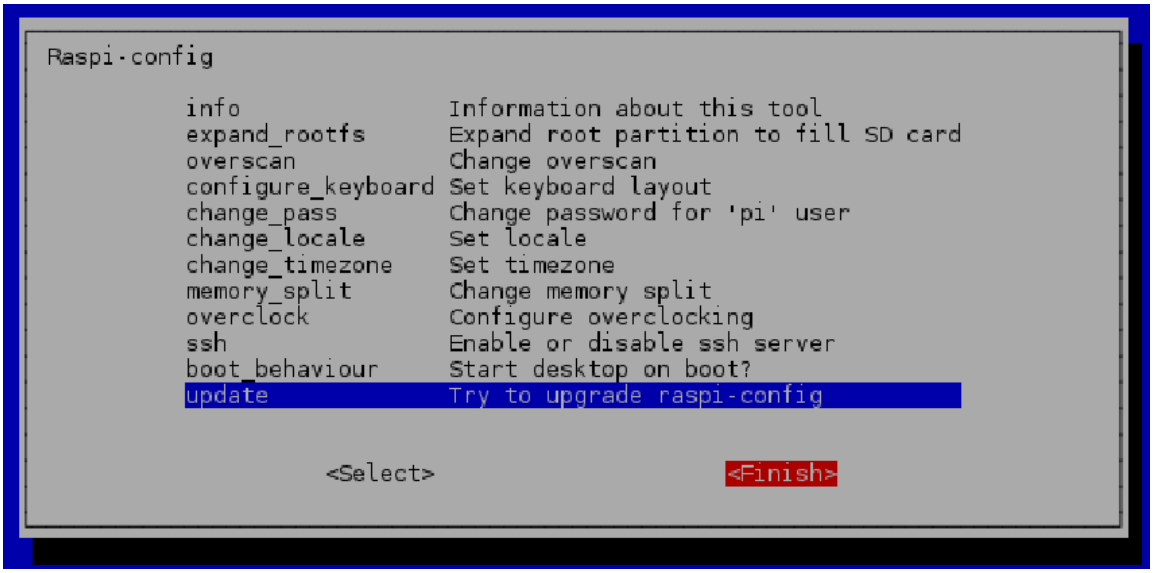

Figura 20– Encerramento das configurações

Visto todos estes passos, e realizados, o sistema operacional está instalado e pronto para o uso.

2.2.4. CONFIGURAÇÕES DE PERSONALIZAÇÃO

Todo o desenvolvimento do projeto é feito em ambiente texto, e o editor de texto utilizado foi o Mcedit.

Atualização do software e instalação do editor de texto Mcedit:

apt-get update	Realiza a atualização do sistema.
apt-get install mc	Inicializa a instalação do editor de texto mcedit.

Configurando os atalhos do profile Pi:

cd /root/etc	Entra no diretório /root/etc
mcedit profile	Através do mcedit abre o arquivo profile
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=1000	Configura o tamanho max. do history
export HISTFILESIZE=1000	

Alterando as configurações do root:

cd /root	Entra no diretório /root
mcedit .bashrc	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=1000	Configura o tamanho max. do history
export HISTFILESIZE=1000	

Para configurar tela cheia:

cd /boot	Entra no diretório /boot
mcedit config.txt	Abre o arquivo para edição

Descomente tirando o caracter '#' antes de: "disable_overscan=1"
Deve-se dar reboot para aplicar as mudanças.

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

mcedit /etc/inittab
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>&

Deve-se dar reboot para aplicar as mudanças.

2.2.5. BIBLIOTECA BCM2835

Para a utilização dos periféricos da GPIO RaspberryPi visto na figura 3, e outras funções do chip da Broadcom, é necessário a instalação da biblioteca BCM2835, podendo ser visto a seguir os passos necessários:

```
wget www.lt38c.hturbo.com/bcm2835-1.33.tar.gz -o /tmp/bcm2835.tar.gz
tar –zvxf bcm2835-1.33.tar.gz
cd bcm2835-1.33
./configure
make check
make install
```

2.3. CONCLUSÃO

Neste capítulo foi apresentado as etapas necessárias para a configuração do micro-sd, e também as instalações de programas e bibliotecas necessárias para o desenvolvimento do projeto. Após as instalações e configurações iniciais a *Raspberry Pi* estará pronta para ser utilizada e descarregar o programa principal e suas pertinências.

CAPÍTULO 3

PROGRAMAS CONSTRUÍDOS OU ADAPTADOS

3.1. INTRODUÇÃO

A sigla UART vem do inglês Universal Asynchronous Receiver/Transmitter ou do português “Receptor/Transmissor Assíncrono Universal”. O dispositivo UART executa a conversão serial-paralela quando é um receptor e paralela-serial quando é um transmissor. Resumindo, é responsável pela comunicação de dados paralelos que trafegam por um meio serial. A Figura 15 ilustra esse tipo de comunicação. A conexão entre os terras (GND) do transmissor com o receptor faz-se necessário devido ao fato de que ambos precisam ter a mesma referência para discernir entre nível lógico baixo e nível lógico alto.

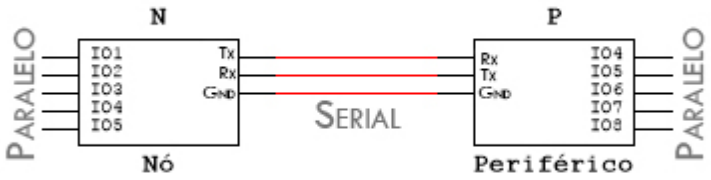

Figura 21– A comunicação UART

Uma comunicação é denominada serial quando o envio dos códigos dos caracteres se processa sobre uma única linha, onde os bits enviados são encadeados em uma fila. É considerada assíncrona quando os relógios (clocks) do transmissor e do receptor não estão sincronizados. A velocidade de comunicação é chamada de baud. Essa unidade representa o número de bits transmitidos por segundo.

Quanto a sincronia dos transmissores e receptores UART pode-se dizer que a cada bit transmitido no canal deve respeitar a temporização de 1/baud segundos. Por exemplo, se for escrito no canal o bit 1, esse canal permanecerá em nível lógico alto por 1/baud segundos.

Através da Figura 16 pode-se observar um exemplo de como a transmissão do caractere W pode ser feita através de dispositivos UART. A conversão binária da letra W é feita com base na tabela ASCII.

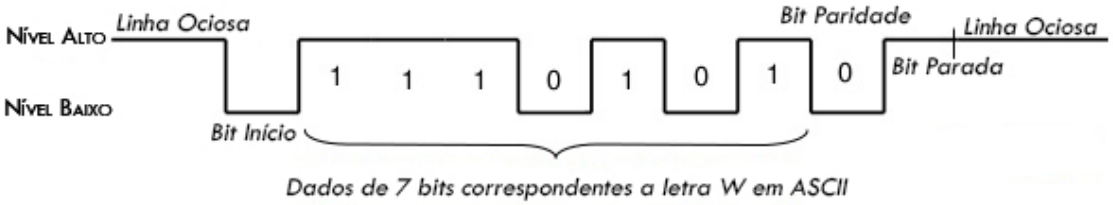

Figura 22– Exemplo de transmissão do caractere W

Os dados são comumente transmitidos de uma ponta a outra em forma de pacotes os quais podem ser tão pequenos quanto um único byte. O formato desses pacotes é baseado em um número conhecido de protocolos. Cada pacote pode incluir uma verificação para detectar se o pacote esta correto ou foi corrompido ao longo da transmissão. Essa verificação poder ser através de checksum, bit de paridade ou Cyclic Redundancy Check Byte (CRC).

Existem três modos que uma comunicação UART pode assumir:

· Modo HALFDUPLEX: Nesse modo a transmissão e a recepção acontecem uma de cada vez.

· Modo FULLDUPLEX: Nesse modo a transmissão e a recepção acontecem ao mesmo tempo.

· Modo LOOP BACK: Esse modo é usado apenas para teste. Acontece quando o pino Tx é conectado ao pino Rx da mesma UART para fins de verificar a funcionalidade da mesma.

3.2. PROGRAMA PRINCIPAL

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
unsigned char pacote[16];
unsigned char recebe[16];
unsigned char entrada;
char comando=0;
int flagMenu=0;
int flagRx=0;
int fd;
int k=0;
int b = 0;
int contador=0;
int exec=0;
int tentativa = 0;
union checksum {
```

```
                unsigned short int soma;
                unsigned short int soma2;
                unsigned char checksumm[2];
        };
        void geraPacote();
        void controleLed(int select);
        void geraSoma();
        void checaSoma();
        void mostraPacote();
        int transmiteUART();
        int kbhit(void);
        int verificaRx();
#define BAUDRATE B9600
#define MODEMDEVICE "/dev/ttyS0"
#define _POSIX_SOURCE 1 /* POSIX compliant source */
#define FALSE 0
#define TRUE 1
        void main(){
                struct termios oldtio,newtio;
                tcgetattr(fd,&oldtio);
                fd = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NONBLOCK);
  tcgetattr(fd,&oldtio);
  newtio.c_cflag = BAUDRATE | CS8 | CLOCAL | CREAD;
  newtio.c_iflag = IGNPAR| IXON | IXOFF;
  newtio.c_oflag = 0;
  newtio.c_lflag = 0;
  newtio.c_cc[VTIME]    = 0;
  tcflush(fd, TCIFLUSH);
  tcsetattr(fd,TCSANOW,&newtio);
                while(comando != 'q'){
                        if(kbhit()){
                                comando = getchar();
                        }
                        if(flagMenu == 0){
                                printf("Digite a opcao desejada: \n a - Acende/Apaga Led 1\n b - Acende/Apaga Led 2\n c - Acende/Apaga Led 3\n");

                                flagMenu=1;
                        }

                        if(comando == 'a'){
                                b = 1;
                                printf("Enviando comando para acender/apagar Led 1\n");
                                controleLed(b);
                                mostraPacote();
                                geraSoma();
                                transmiteUART();
                                comando = 0;
                        }
                        if(comando == 'b'){
                                b = 2;
                                printf("Enviando comando para acender/apagar Led 2\n");
                                controleLed(b);
                                mostraPacote();
                                geraSoma();
                                transmiteUART();
                                comando = 0;
                        }
                                if(comando == 'c'){
                                b = 3;
```

```
                                        printf("Enviando comando para acender/apagar Led 2\n");
                                        controleLed(b);
                                        mostraPacote();
                                        geraSoma();
                                        transmiteUART();
                                        comando = 0;
                                }
                        verificaRx();
                        }
        }
void mostraPacote(){
        int i;
        for(i = 0; i<16 ; i++ ){
                printf("%c",pacote[i]);
        }
        printf("\n\n\n");
}
void geraPacote(){
        pacote[0] = '#';
        int i=1;
        srand( (unsigned)time(NULL) );
        for(i=1;i<16;i++){
                pacote[i] = 65 + rand()%126;
        }
}
void controleLed(int b){
        int a = b;
        geraPacote();
        pacote[1] = 'L';
        pacote[2] = 'E';
        pacote[3] = 'D';
        switch(a){
        case 1:
        pacote[4] = '1';
        break;
        case 2:
        pacote[4] = '2';
        break;
        case 3:
        pacote[4] = '3';
        break;
                default:
                printf("Não tem esse led fera!\n");
}
        }
        void geraSoma(){
                int j=0;
                union checksum chk;
                chk.soma = 0;
                for(j=0; j<=13; j++){
                        chk.soma += pacote[j];
                }
                pacote[14] = chk.checksumm[0];
                pacote[15] = chk.checksumm[1];
                printf("A soma total eh: %d\n ",chk.soma);
                printf("A soma 1  eh: %d\n ",pacote[14]);
                printf("A soma 2 eh: %d\n\n ",pacote[15]);
        }
        int transmiteUART(){
                int n=0;
```

```
		if(flagRx==0){

			if (fd == -1) {
				perror("Não deu pra abrir a porta serial /dev/ttyAMA0 - \n ");
				return(-1);
			}
			fcntl(fd, F_SETFL, 0);
			n = write(fd,&pacote[0],16);
			flagRx = 1;
			if (n < 0) {
				perror("Falha na escrita \n");
				return -1;
			}
			printf("transmiti\n\n");
		}
		return 0;
	}
	int kbhit(void){
		struct termios oldt, newt;
		int ch;
		int oldf;
		tcgetattr(STDIN_FILENO, &oldt);
		newt = oldt;
		newt.c_lflag &= ~(ICANON | ECHO);
		tcsetattr(STDIN_FILENO, TCSANOW, &newt);
		oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
		fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);
		ch = getchar();
		tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
		fcntl(STDIN_FILENO, F_SETFL, oldf);
		if(ch != EOF){
			ungetc(ch, stdin);
			return 1;
		}
		return 0;
}
	int verificaRx(){
		int n=0;
		if(flagRx==1){
			contador++;
			n = read(fd,&entrada,1);
			if(n ){
				recebe[0] = recebe[1];
				recebe[1] = recebe[2];
				recebe[2] = recebe[3];
				recebe[3] = recebe[4];
				recebe[4] = recebe[5];
				recebe[5] = recebe[6];
				recebe[6] = recebe[7];
				recebe[7] = recebe[8];
				recebe[8] = recebe[9];
				recebe[9] = recebe[10];
				recebe[10] = recebe[11];
				recebe[11] = recebe[12];
				recebe[12] = recebe[13];
				recebe[13] = recebe[14];
				recebe[14] = recebe[15];
				recebe[15] = entrada;
				k++;
			}
```

```
                        if(recebe[0] == '#'){
                                printf("to exec...");
                                checaSoma();
                                recebe[0] = 0;
                                printf(    "fui chamado %d vezes",k);
                        }
                         if(contador>=50000){
                                exec++;
                                contador = 0;
                                printf(    "estourou %d vezes\n",exec);
                         }
                         if(exec>=3){
                                 flagRx=0;
                                 exec=0;
                                 printf("Retransmitindo último pacote...\n");
                                 controleLed(b);
                                 mostraPacote();
                                 geraSoma();
                                 printf("Retransmitindo pacote\n");
                                 transmiteUART();
                                 tentativa++;
                         }
                         if(tentativa>=3){
                                 printf("Impossível conectar ao cliente, verifique as conexões\n FLW
VLW\n");
                                 comando='q';
                         }
                }
return 0;
        }
        void checaSoma(){
                int i=0;
                int j =0;
                union checksum checaRx;
                checaRx.soma = 0;
                checaRx.checksumm[0] = 0;
                checaRx.checksumm[1] = 0;
                printf("Arduino retornou o pacote: ");
                recebe[15] = entrada;
                for(j = 0; j<16 ; j++ ){
                        printf("%c",recebe[j]);
    }
    printf("\n\n\n");
                for(i=0; i<14; i++){
                        checaRx.soma += recebe[i];
                }
                if(recebe[14] == checaRx.checksumm[0] && recebe[15] == checaRx.checksumm[1]){ // se
bater o checksum
                        printf("Recepcao Correta\n");

                        if(recebe[1]=='E' && recebe[2]=='R' && recebe[3]=='R' && recebe[4]=='O'){
                                printf("Erro, o cliente não pode executar o comando enviado!\n");
                        }
                        if(recebe[1]==pacote[1] && recebe[2]==pacote[2] && recebe[3]==pacote[3] &&
recebe[4]==pacote[4] && recebe[5]=='O' && recebe[6]=='K'){
                                printf("Comando enviado foi executado com sucesso!\n");
                                printf("Deu bom\n");
                        printf("check recebido é: %d e %d e o check calculado é %d e %d\n\n", recebe[14],
recebe[15], checaRx.checksumm[0], checaRx.checksumm[1]);
                                flagRx = 0;
```

```
                        flagMenu=0;
                }
                else{
                        printf("Deu ruim, nao veio erro nem OK\n");
                        flagRx = 0;
                        controleLed(b);
                        mostraPacote();
                        geraSoma();
                        printf("Retransmitindo pacote\n");
                        transmiteUART();
                }
        }
        else{
                printf("Deu ruim, não bateu o checksum do pacote recebido\n");
                printf("check recebido é: %d e %d e o check calculado é %d e %d\n\n", recebe[14],
recebe[15], checaRx.checksumm[0], checaRx.checksumm[1]);
        }
}
```

3.2.1. PROGRAMA ADICIONAL

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
const int selectPin1 = 2;
const int selectPin2 = 3;
const int LedPin1 = 8;
const int LedPin2 = 9;
const int LedPin3 = 10;
const int ledPin11 = 11;
const int ledPin12 = 12;
const int ledPin13 = 13;
int selectMSB = 0;
int selectLSB = 0;
const int pinL1 = 4;
const int pinL2 = 5;
const int pinH1 = 6;
const int pinH2 = 7;
unsigned char rec[16];
unsigned char sen[16];
unsigned char ans[2];
unsigned char last;
union checksum{
  unsigned short int soma;
  unsigned char checksum[2];
};
void SerialUART();
void checaSoma();
void checaAciona();
void geraSen();
void deuErrado();
void deuCerto();
void setup(){
        Serial.begin(9600);
        randomSeed(analogRead(0));
        pinMode(pinL1, OUTPUT);
        pinMode(pinL2, OUTPUT);
```

```
        pinMode(pinH1, OUTPUT);
        pinMode(pinH2, OUTPUT);
        digitalWrite(pinL1, LOW);
        digitalWrite(pinL2, LOW);
        digitalWrite(pinH1, HIGH);
        digitalWrite(pinH2, HIGH);
        pinMode(LedPin1, OUTPUT);
        pinMode(LedPin2, OUTPUT);
        pinMode(LedPin3, OUTPUT);
        digitalWrite(LedPin1, LOW);
        digitalWrite(LedPin2, LOW);
        digitalWrite(LedPin3, LOW);
        pinMode(ledPin11, OUTPUT);
        pinMode(ledPin12, OUTPUT);
        pinMode(ledPin13, OUTPUT);
        digitalWrite(ledPin11, LOW);
        digitalWrite(ledPin12, LOW);
        digitalWrite(ledPin13, LOW);
        pinMode(selectPin1, INPUT);
        pinMode(selectPin2, INPUT);
}
void loop(){
        selectLSB = digitalRead(selectPin1);
        selectMSB = digitalRead(selectPin2);
        if(selectLSB == LOW && selectMSB == LOW){
                digitalWrite(ledPin11, HIGH);
                digitalWrite(ledPin12, LOW);
                digitalWrite(ledPin13, LOW);
                SerialUART();
        }
}
void SerialUART(){
        if(Serial.available()){
                last = 0;
                last = Serial.read();
                rec[0] = rec[1];
                rec[1] = rec[2];
                rec[2] = rec[3];
                rec[3] = rec[4];
                rec[4] = rec[5];
                rec[5] = rec[6];
                rec[6] = rec[7];
                rec[7] = rec[8];
                rec[8] = rec[9];
                rec[9] = rec[10];
                rec[10] = rec[11];
                rec[11] = rec[12];
                rec[12] = rec[13];
                rec[13] = rec[14];
                rec[14] = rec[15];
                rec[15] = last;
                last = 0;
        }
        if(rec[0]=='#'){
                checaAciona();
                rec[0] = 0;
        }
}
void checaAciona(){
        union checksum chk;
```

```
	int i;
	chk.soma = 0;
	for(i=0; i<=13; i++){
		chk.soma += rec[i];
	}
	ans[0] = chk.checksum[0];
	ans[1] = chk.checksum[1];
	if(ans[0] == rec[14] && ans[1] == rec[15]){
		if(rec[1] == 'L' && rec[2] == 'E' && rec[3] == 'D'){
			int led;
			if(rec[4] == '1'){
				led = digitalRead(LedPin1);
				if(led == HIGH){
					digitalWrite(LedPin1, LOW);
				}else{
					digitalWrite(LedPin1, HIGH);
				}
			}else if(rec[4] == '2'){
				led = digitalRead(LedPin2);
				if(led == HIGH){
					digitalWrite(LedPin2, LOW);
				}else{
					digitalWrite(LedPin2, HIGH);
				}
			}else if(rec[4] == '3'){
				led = digitalRead(LedPin3);
				if(led == HIGH){
					digitalWrite(LedPin3, LOW);
				}else{
					digitalWrite(LedPin3, HIGH);
				}
			}
			deuCerto();
		}else{
			deuErrado();
		}
	}else{
		deuErrado();
	}
}
void deuCerto(){
	int i;
	geraSen();
	sen[1] = rec[1];
	sen[2] = rec[2];
	sen[3] = rec[3];
	sen[4] = rec[4];
	sen[5] = 'O';
	sen[6] = 'K';
	checaSoma();
	for(i=0; i<=15; i++){
		Serial.write(sen[i]);
	}
}
void deuErrado(){
	int i;
	geraSen();
	sen[1] = 'E';
	sen[2] = 'R';
	sen[3] = 'R';
```

```
		sen[4] = 'O';
		checaSoma();
		for(i=0; i<=15; i++){
			Serial.write(sen[i]);
		}
}
void geraSen(){
		int i;
		sen[0] = '#';
		for(i=1; i<=15; i++){
			sen[i] = random(65,126);
		}
}
void checaSoma(){
		union checksum chk2;
		int i;
		chk2.soma = 0;
		for(i=0; i<=13; i++){
			chk2.soma += sen[i];
		}
		sen[14] = chk2.checksum[0];
		sen[15] = chk2.checksum[1];
}
```

3.3. FLUXO DAS INFORMAÇÕES

Primeiramente a Raspberry cria um pacote de dados, para o acionamento de um LED, e envia o mesmo para o microcontrolador presente na Gertboard. Logo que recebe o pacote o microcontrolador verifica se o mesmo foi recebido corretamente, aciona o LED, cria e envia outro pacote para informar que o LED foi acionado. Ao receber o pacote, a raspberry verifica se o LED foi acionado e aguarda novos comandos.

3.4. ESTRUTURA DOS DIRETÓRIOS DO PROJETO

- 2015_2s_serial
  - arquivos
  - exe
  - fontes.c
  - history
  - maff.mht
  - mysql.sql
  - pacotes
  - site
  - videos

3.5. CONCLUSÃO

Para o funcionamento correto da comunicação, os dois códigos devem estar devidamente compilados e gravados respectivamente na Raspberry e no microcontrolador da Gertboard.

CAPÍTULO 4

DESCRIÇÃO DO HARDWARE ADICIONAL NA GPIO

4.1. INTRODUÇÃO

Para o desenvolvimento do trabalho foi utilizada a GERTBOARD, assim é possível realizar a comunicação entre a Raspberry Pi e o microcontrolador presente na placa auxiliar.

4.2. GERTBOARD

Um dispositivo sancionado Raspberry Pi Foundation concebidos para fins educativos, e expande pinos GPIO do Raspberry Pi para permitir interação eo controle de LEDs, interruptores, sinais analógicos, sensores e outros dispositivos. Ele também inclui um controlador compatível com Arduino para fazer a interface com o Pi.

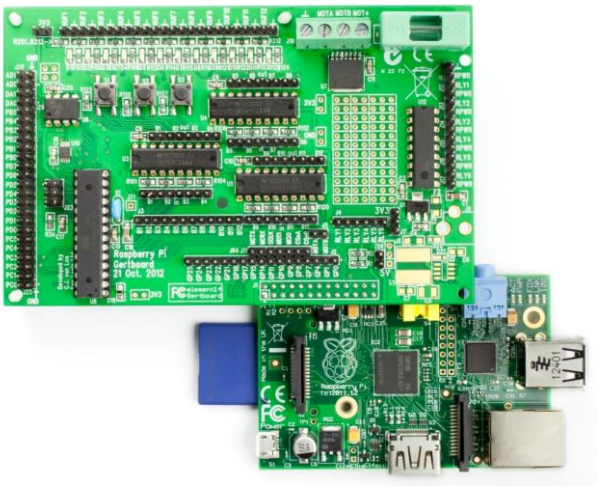

Figura 23 - Visão geral Gertboard e Raspberry Pi

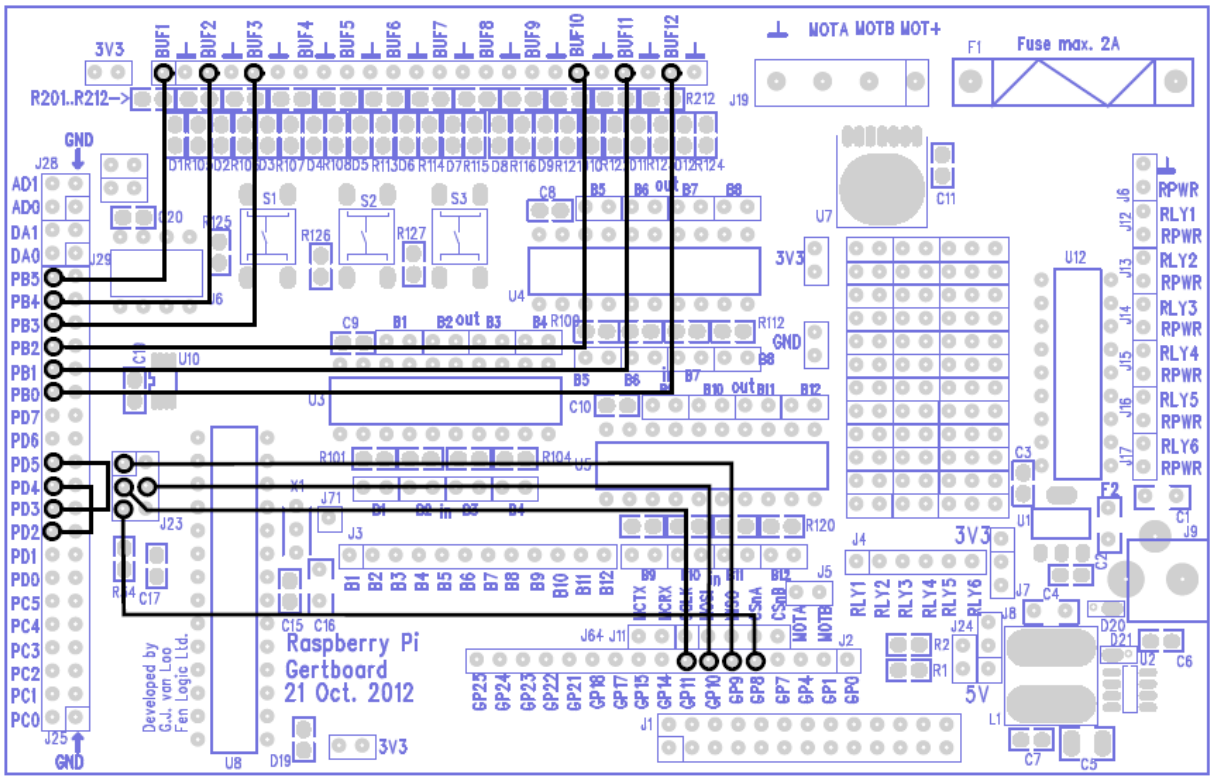

Figura 24 - Ligação dos pinos na GERTBOARD.

REFERÊNCIAS BIBLIOGRÁFICAS

TANENBAUM, A. S. **Redes de Computadores.** Rio de Janeiro: Campus, 1997.

CURRIE, E. H.; ESS, D. V. **PSoC3/5 Reference Book**. [S.l.]: Cypress Semiconductor Corporation, 2010.

KATHIK, T. K. *et al*. Design and verification of uart ip core using vmm. **International Journal of Soft Computing and Engineering**, v. 2, 2012.

KRISHNAKISHORE, G.; SHRUTI, K.; VARSHA, M. Design and simulation of i2c bus using verilog. **International Journal of Engineering Trends and Technology**, v. 4, 2014.

KUMAR, M. A.; SNEHA, B. Video acquisition through i 2 c using vhdl. **International Journal of Engineering Trends and Technology**, v. 4, 2013.

SACCO, F. **Comunicação SPI**. 2014. Disponível em: <http://www.embarcados.com.br/spi-parte-1/>. SAMANTA, A. G. Interfacing of pic 18f252 microcontroller with real time clock via i2c protocol. **Int. Journal of Engineering Research and Applications**, v. 4, p. 152–156, 2014.

SANDYA, M.; RAJASEKHAR, K. Design and verification of serial peripheral interface. **International Journal of Engineering Trends and Technology 3 (4)**, p. 522–524, 2012.