UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CAMPO MOURÃO

GERÊNCIA DE PESQUISA E GRADUAÇÃO

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

LEONARDO FERNANDO WALKER, CAIKE RODRIGO ALBERTIN.

PROJETO SOCKET

PROJETO FINAL DE DISCIPLINA

CAMPO MOURÃO, JULHO 2015

LEONARDO FERNANDO WALKER, CAIKE RODRIGO ALBERTIN.

PROJETO SOCKET

Trabalho de conclusão de disciplina apresentada ao Professor de Sistemas Embarcados no curso de Graduação em Engenharia Eletrônica 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, JULHO 2015

Sumário

CAPÍTULO 1 1
1. INTRODUÇÃO 1
1.1 SISTEMAS EMBARCADOS 1
1.2 RASPBERRY PI 1
1.3 MOTIVAÇÕES 1
1.4 OBJETIVOS 1
CAPÍTULO 2 2
INSTALAÇÃO DO SISTEMA INICIAL 2
2.1 INTRODUÇÃO 2
2.2 DESCRIÇÃO SUSCINTA DAS INTALAÇÕES 2
2.3 BIBLIOTECA BCM2835 8
2.4 CONCLUSÃO 9
CAPÍTULO 3 10
PROGRAMAS CONSTRUÍDOS OU ADAPTADOS 10
3.1 INTRODUÇÃO 10
3.2 PROGRAMA PRINCIPAL 10
3.2.1 PROGRAMA ADICIONAL 13
3.3 FLUXO DAS INFORMAÇÕES 14
3.4 CONCLUSÃO 14
CAPÍTULO 4 15
DESCRIÇÃO DO HARDWARE ADICIONAL NA GPIO 15
4.1 INTRODUÇÃO 15
4.2 SERVOMOTOR 15
4.4 CONCLUSÃO 16
REFERÊNCIAS BIBLIOGRÁFICAS 17

Figura 1 – Conexões da raspberry Pi 2
Figura 2 – Expansão do micro-sd 3
Figura 3 – Configuração teclado 3
Figura 4 – Configuração do teclado 4
Figura 5 – Configuração do teclado 4
Figura 6 – Configuração do teclado 4
Figura 7 – Configuração do teclado 4
Figura 8 – Configuração da localização 5
Figura 9 – Configuração da localização 5
Figura 10 – Configuração da localização 5
Figura 11 – Configuração da timezone 6
Figura 12 – Configuração da timezone 6
Figura 13 – Configuração da timezone 7
Figura 14 – Encerramento das configurações 7
Figura 15 – Controle da posição de um servomotor 15
Figura 16 – Conexões do servomotor ao Raspberry Pi 16

LISTA DE ABREVIATURAS E SIGLAS

GNU - Sistema Operacional do tipo Unix

GPIO - *General Purpose Input/Output*

RISC - *Reduced Instruction Set Computer*

SO - Sistema Operacional

PWM - *Pulse width Modulation.*

RESUMO

O projeto consiste em criar um servidor que seja capaz de controlar por PWM um servomotor a partir de um comando enviado por um cliente localmente conectado. Sabe-se que a raspberry Pi possui saída PWM, utilizando a biblioteca BCM2835 controla-se o servomotor. Utilizou-se socket para realizar a comunicação entre cliente e servidor, onde o cliente manda um comando e o servidor realiza uma ação, alterando a largura de pulso do PWM de saída.

CAPÍTULO 1

1. INTRODUÇÃO

1.1 SISTEMAS EMBARCADOS

Um sistema embarcado é um sistema microprocessado, onde o computador é completamente dedicado ao dispositivo ou ao sistema que ele controla. 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.

1.2 RASPBERRY PI

Raspberrry pi pode ser considerado um computador portátil que pode-se conectar a periféricos, como um monitor, um teclado e um mouse, foi desenvolvido no reino unido pela fundação raspberry pi. Foi lançada no início de 2012. Seu objetivo principal é promover o ensino de ciências da computação básica nas escolas. Ela se caracteriza por ter um preço acessível.

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.

1.4 OBJETIVOS

O objetivo deste trabalho é gerar uma comunicação local do tipo cliente servidor via socket, para o controle de posição de um servomotor.

CAPÍTULO 2

INSTALAÇÃO DO SISTEMA INICIAL

2.1 INTRODUÇÃO

O sistema Linux que será utilizado é o Raspbian, 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 INTALAÇÕES

Inicia-se a instalação no micro-sd, de pelo menos 4gb, com o sistema Raspbian encontrado no site https://www.raspberrypi.org/downloads.

É necessário fazer o download do programa *Win32DiskImager*, que é o programa para fazer a gravação do sistema operacional no micro-sd. A figura 1 contem exemplos de conexão que podem ser realizadas à placa do Raspberry Pi.

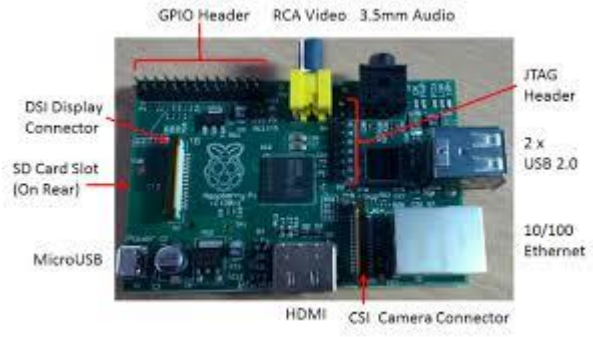

Figura 1 – Conexões da raspberry Pi

Após a instalação, é necessário fazer as configurações do sistema operacional, tal como horário, teclado, entre outros.

O próximo passo é expandir o micro-sd de modo a possibilitar a utilização de seu tamanho total.

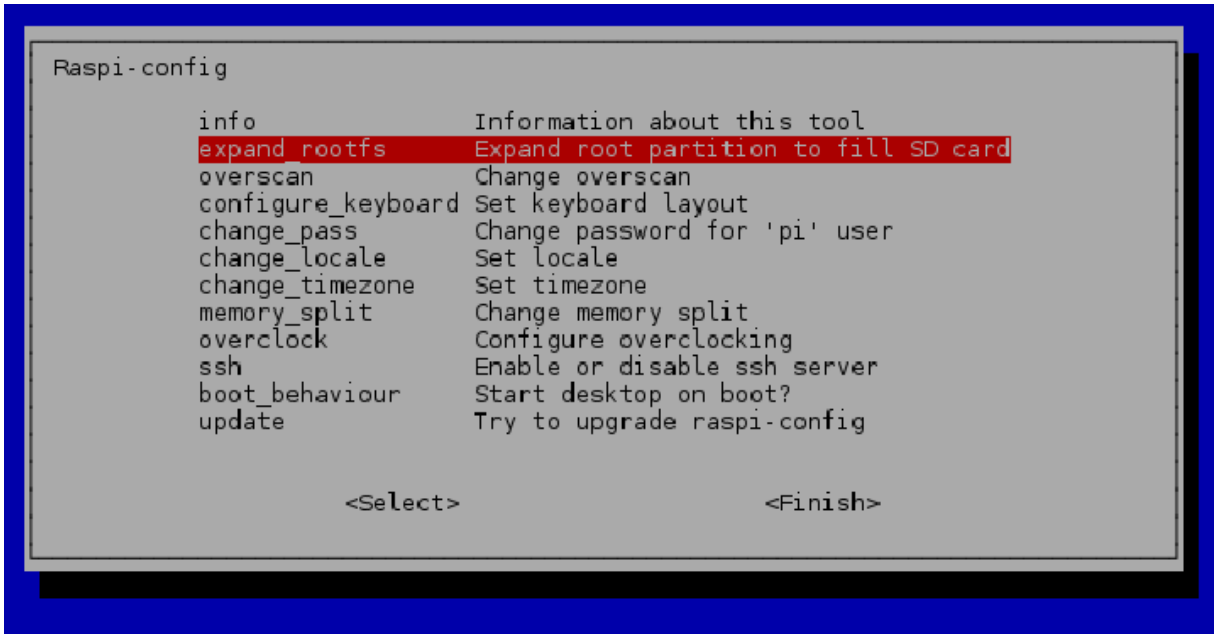

Figura 2 – Expansão do micro-sd

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.

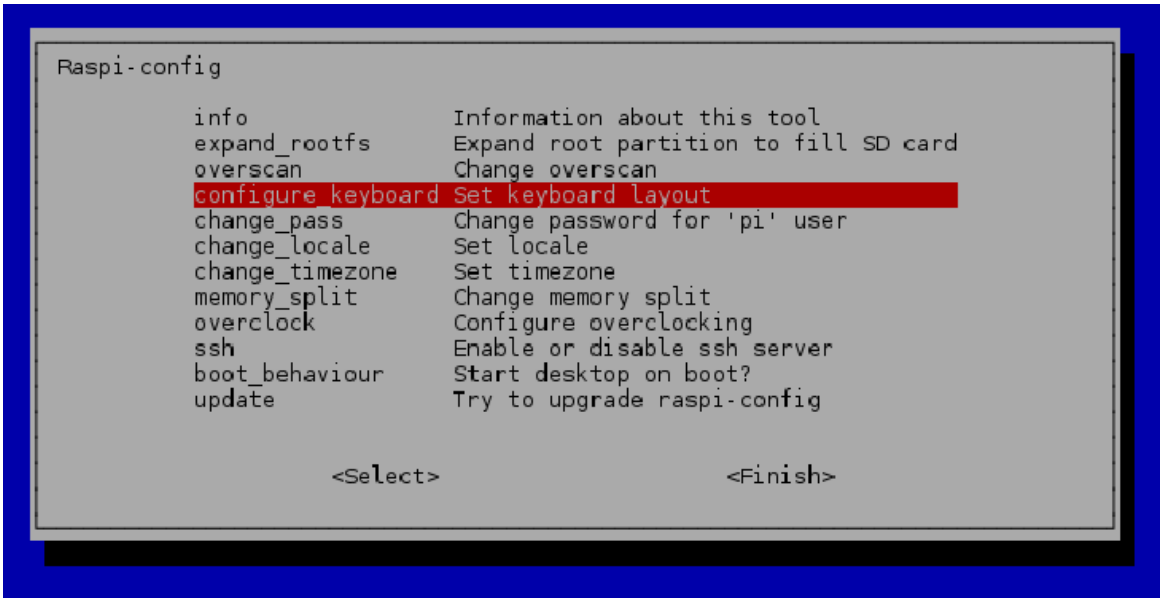

Figura 3 – Configuração teclado

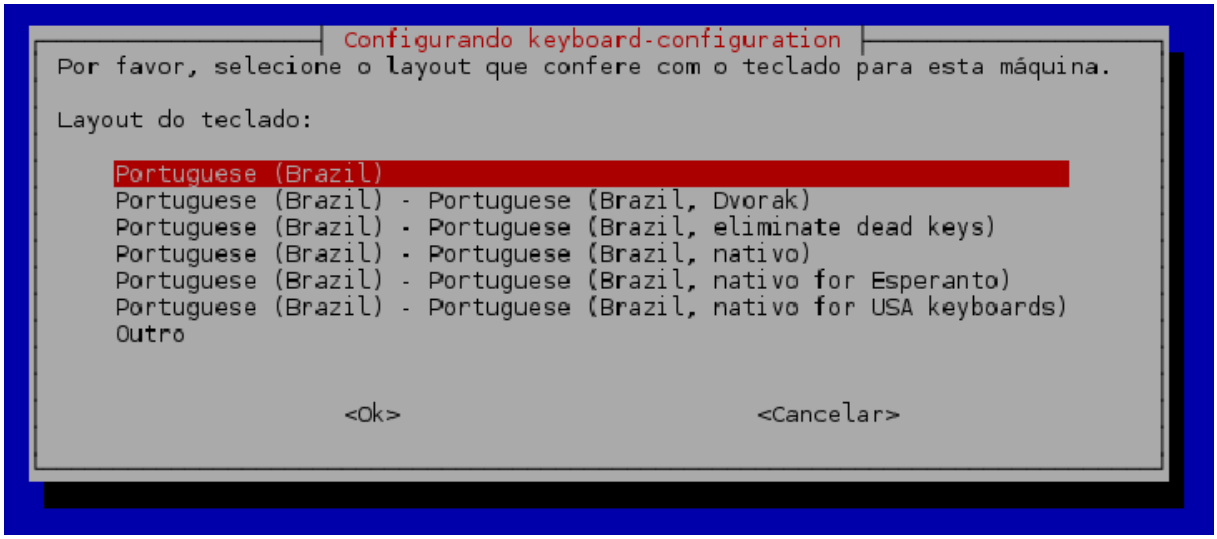

Figura 4 – Configuração do teclado

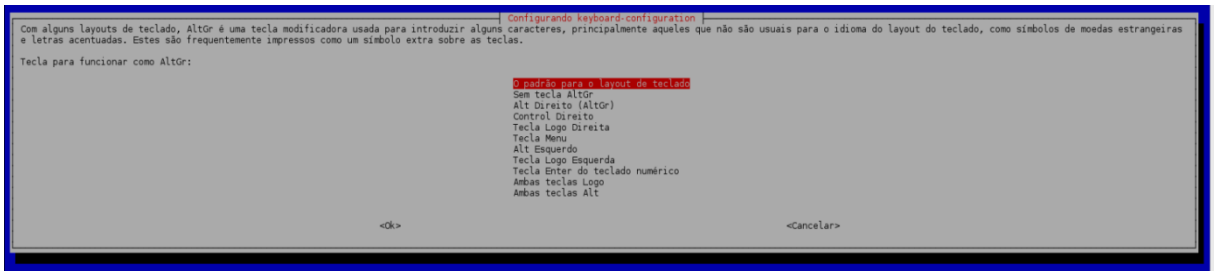

Figura 5 – Configuração do teclado

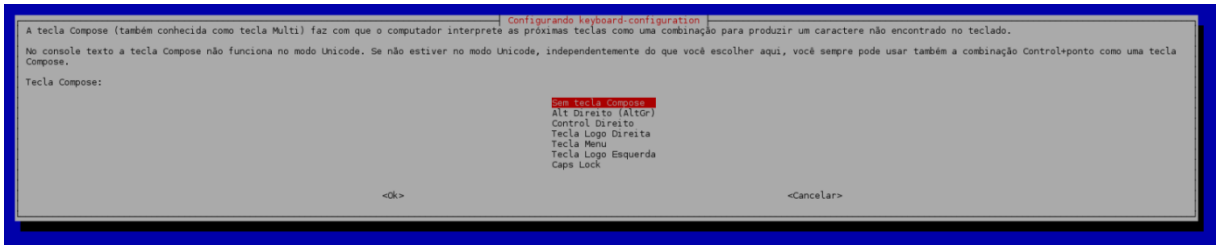

Figura 6 – Configuração do teclado

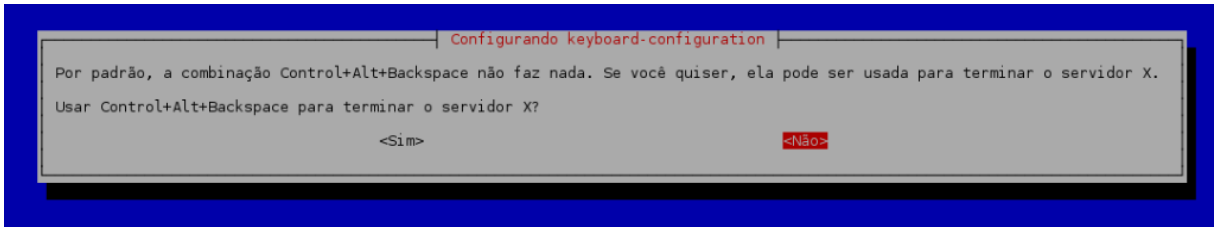

Figura 7 – Configuração do teclado

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

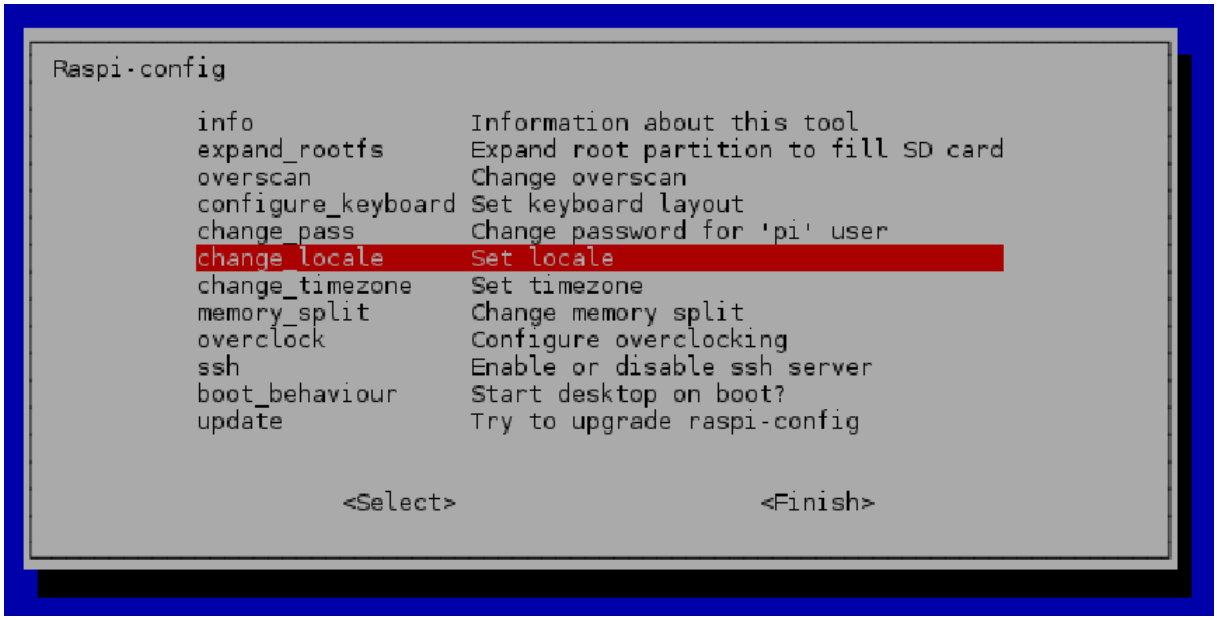

Figura 8 – Configuração da localização

Figura 9 – Configuração da localização

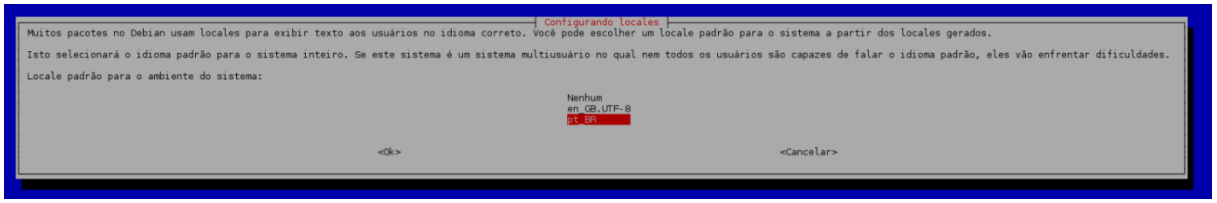

Figura 10 – Configuração da localização

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.

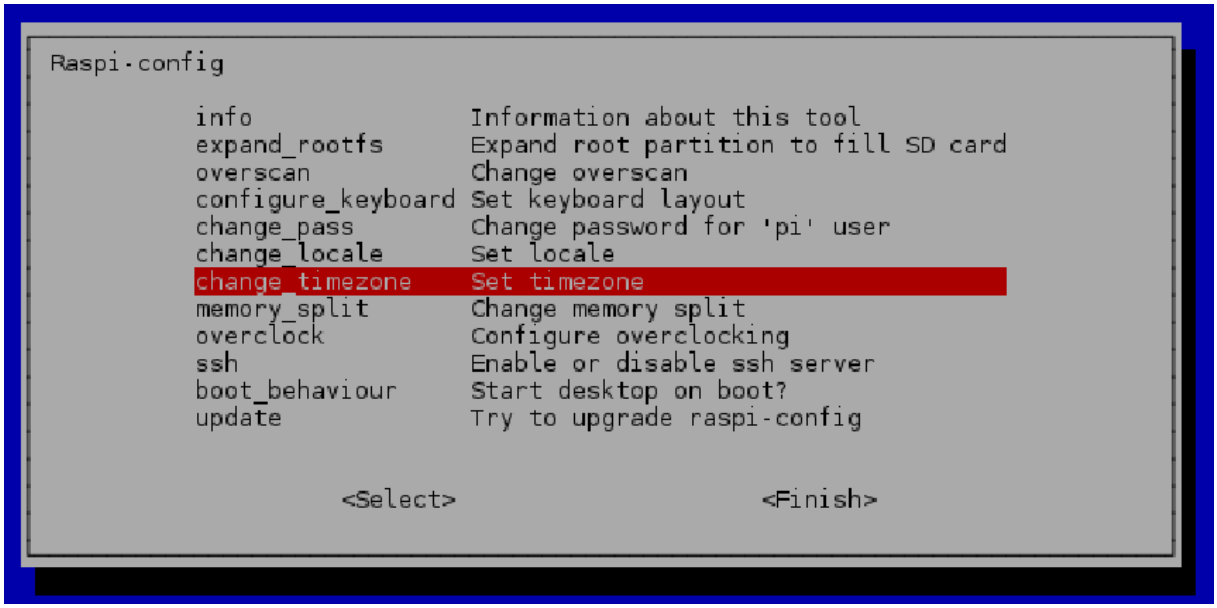

Figura 11 – Configuração da timezone

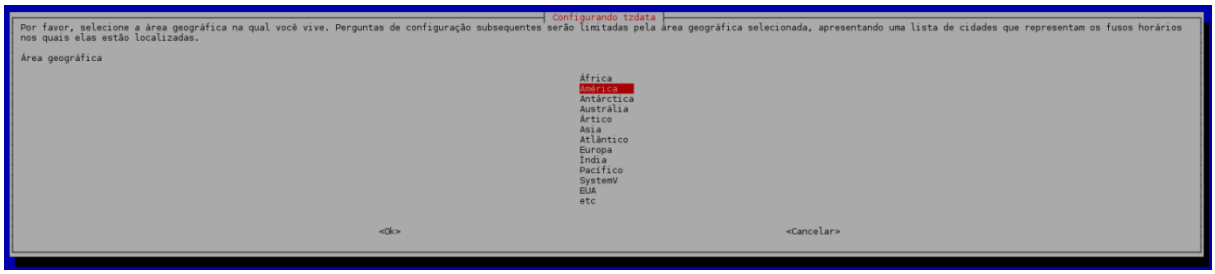

Figura 12 – Configuração da timezone

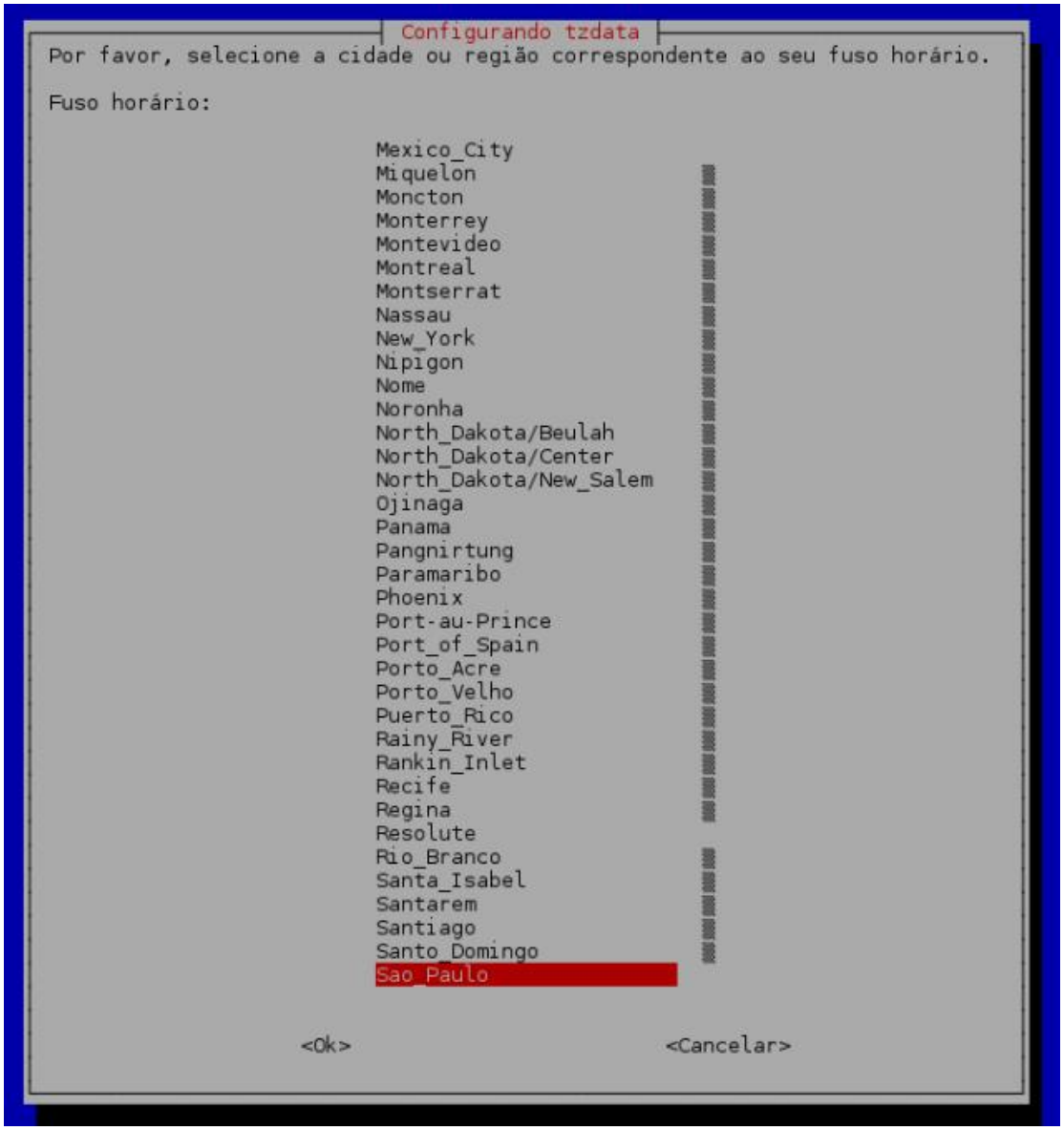

Figura 13 – Configuração da timezone

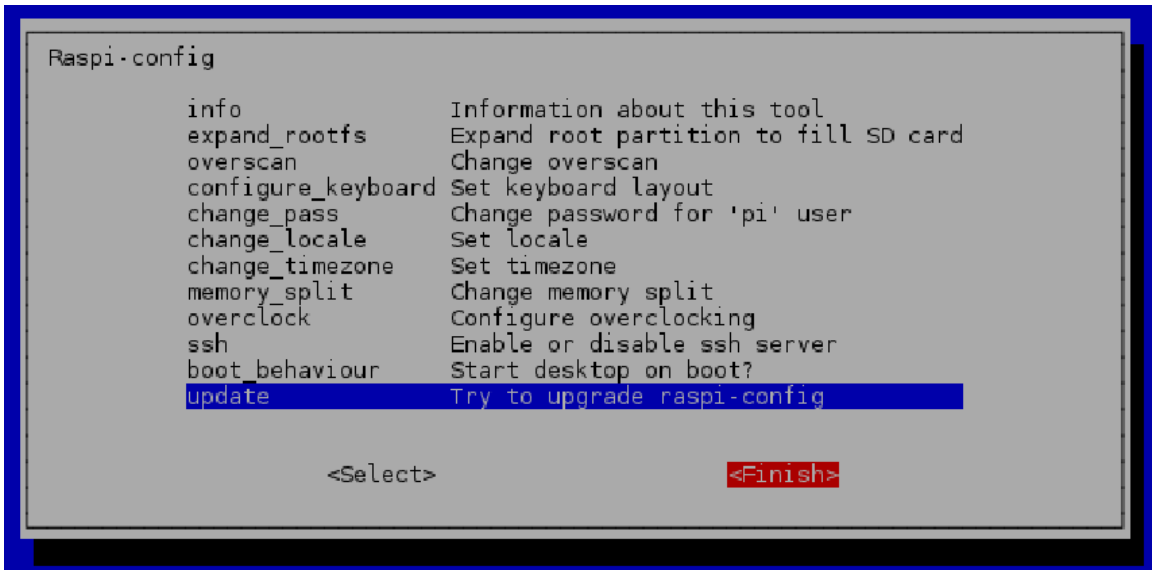

Figura 14 – Encerramento das configurações

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"
```

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>&1
```

2.2.1 BCM 2835-1.43

Tanto para configurar o root e pi, quanto configurar tela cheia e login deve-se dar reboot para aplicar as mudanças

2.3 BIBLIOTECA BCM2835

É uma biblioteca em C para a Raspberry, possui comandos para controlar entradas e saídas (GPIO), contendo acesso e facilitando o acesso dos pinos. Ela fornece funções que

permitem controlar um PWM, necessário para o controle da posição do servomotor. Para efetuar a instalação é necessário seguir os passos abaixo:

```
wget www.lt38c.hturbo.com/bcm2835-1.43.tar.gz -O /tmp/bcm2835.tar.gz
cd/tmp
tar –zvxf bcm2835-1.43.tar.gz
cd bcm2835-1.43
./configure
make
sudo make check
sudo make install
```

Para usar a biblioteca em C deve-se digitar #include <bcm2835.h> no arquivo.c.

2.4 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.

CAPÍTULO 3

PROGRAMAS CONSTRUÍDOS OU ADAPTADOS

3.1 INTRODUÇÃO

O intuito deste projeto é utilizar as ferramentas fornecidas pelo socket para a comunicação local do tipo cliente-servidor. As funções serão detalhadas ao decorrer do capítulo.

3.2 PROGRAMA PRINCIPAL

```
#include<stdio.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<strings.h>
#include<string.h>
#include<unistd.h>
#include<netdb.h>
#include<errno.h>
#include<bcm2835.h>
int main(void){
int iSock;
struct sockaddr_in server_addr;
int X;
X=1;
iSock=socket(AF_INET, SOCK_STREAM, 0);
if(iSock<0){
printf("Erro ao criar socket\n");
exit(1);
}
server_addr.sin_family = AF_INET;
```

```
server_addr.sin_port = htons(25000);
server_addr.sin_addr.s_addr = INADDR_ANY;
bzero(&(server_addr.sin_zero),8);
if(bind(iSock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr))==-1){
printf("Erro ao executar bind\n");
exit(1);
}
if(listen(iSock, 10)<0){
printf("Erro ao escutar a porta 25000\n");
exit(1);
}
if(!bcm2835_init()){
printf("Erro no Pwm");
return 1;
}
bcm2835_gpio_fsel(RPI_GPIO_P1_12, BCM2835_GPIO_FSEL_ALT5);
bcm2835_pwm_set_clock(375);
bcm2835_pwm_set_mode(0, 1, 1);
bcm2835_pwm_set_range(0, 1024);
while(X==1){
printf("Procurando cliente...\n");
struct sockaddr_in cliente_addr;
int iFd;
int num;
int sin_size;
sin_size = sizeof(struct sockaddr_in);

if((iFd=accept(iSock, (struct sockaddr*)&cliente_addr, &sin_size))<0){
perror("Erro ao estabelecer conexao\n");
exit(1);
}
printf("Cliente conectado: %s\n", inet_ntoa(cliente_addr.sin_addr));
int Bytes_rec;
char buffer_[200];
```

```
char A_[]="A";
char B_[]="B";
char C_[]="C";
if(Bytes_rec = recv(iFd, buffer_, 200, 0)<0){
printf("Erro ao receber mensagem do cliente");
close(iFd);
exit(1);
}
if(buffer_[0] == 'A'){
printf("O angulo escolhido eh 0 graus\n");
bcm2835_pwm_set_data(0, 51);
}
else{
   if(buffer_[0] == 'B'){
   printf("O angulo escolhido eh 90 graus\n");
   bcm2835_pwm_set_data(0, 78);
   }
   else{
         if(buffer_[0] == 'C'){
         printf("O angulo escolhido eh 180 graus\n");
         bcm2835_pwm_set_data(0, 104);
         }
         else{
            if(buffer_[0] == 'S'){
            printf("O servidor terminou sua execucao\n");
            X=0;
            }
            else{
            printf("Comando invalido\n");
            printf("Por favor escolha um comando entre os citados: %s %s %s\n", A_,
B_, C_);
            }
         }
   }
```

```
}
close(iFd);
}
}
```

3.2.1 PROGRAMA ADICIONAL

```
#include<stdio.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<strings.h>
#include<string.h>
#include<unistd.h>
#include<netdb.h>
#include<errno.h>
int main(int argc, char *argv[]){
struct sockaddr_in server_addr;
int iSock;
char num[200];
iSock = (socket(AF_INET, SOCK_STREAM, 0));
if(iSock < 0){
perror("Erro ao criar socket");
exit(1);
}
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(25000);
server_addr.sin_addr.s_addr = INADDR_ANY;
bzero(&(server_addr.sin_zero), 8);
if(connect(iSock, (struct sockaddr*)&server_addr, sizeof(struct sockaddr))<0){
perror("Erro ao conectar\n");
exit(1);
}
int iBytes;
printf("Digite\n A - Para 0 graus\n B - Para 90 graus\n C - Para 180 graus\n S - Para Terminar
a execucao\n");
scanf("%s", num);
printf("Conectado.\n");
if(send(iSock, num, 200, 0)<0){
perror("Erro ao enviar mensagem\n");
exit(1);
}
printf("A opcao escolhida foi: %s\n", num);
send(iSock, num, 200, 0);
close(iSock);
}
```

3.3 FLUXO DAS INFORMAÇÕES

O servidor fica esperando conexão local com um cliente e quando é realizada a conexão o cliente gera um menu de opções, fazendo a escolha de uma das opções (0º, 90º e 180º) e envia um char contendo uma das opções para o servidor, a partir disso é moldado o ciclo de trabalho do pwm de saída através de funções da biblioteca bcm2835.

3.4 CONCLUSÃO

Com a utilização das funções send e recv, é possível enviar e receber dados, tanto localmente quanto globalmente, ou seja, utilizando corretamente as estruturas, é possível se comunicar com qualquer computador na rede.

CAPÍTULO 4

DESCRIÇÃO DO HARDWARE ADICIONAL NA GPIO

4.1 INTRODUÇÃO

Servomotor é uma máquina, eletromecânica, que apresenta movimento proporcional a um comando. Servomotores são dispositivos de malha fechada, ou seja: recebem um sinal de controle; que verifica a posição atual para controlar sua movimentação indo para a posição desejada com velocidade monitorada externamente sob feedback de um dispositivo denominado taco ou sensor de efeito Hall ou encoder ou resolver, ou tachsin, dependendo do tipo de servomotor e aplicação.

4.2 SERVOMOTOR

A posição do servomotor pode ser controlada por PWM, no qual a verificação da posição é feita em intervalos de 20ms. A figura 15 contém os PWMs com as larguras de pulsos pré-definidas.

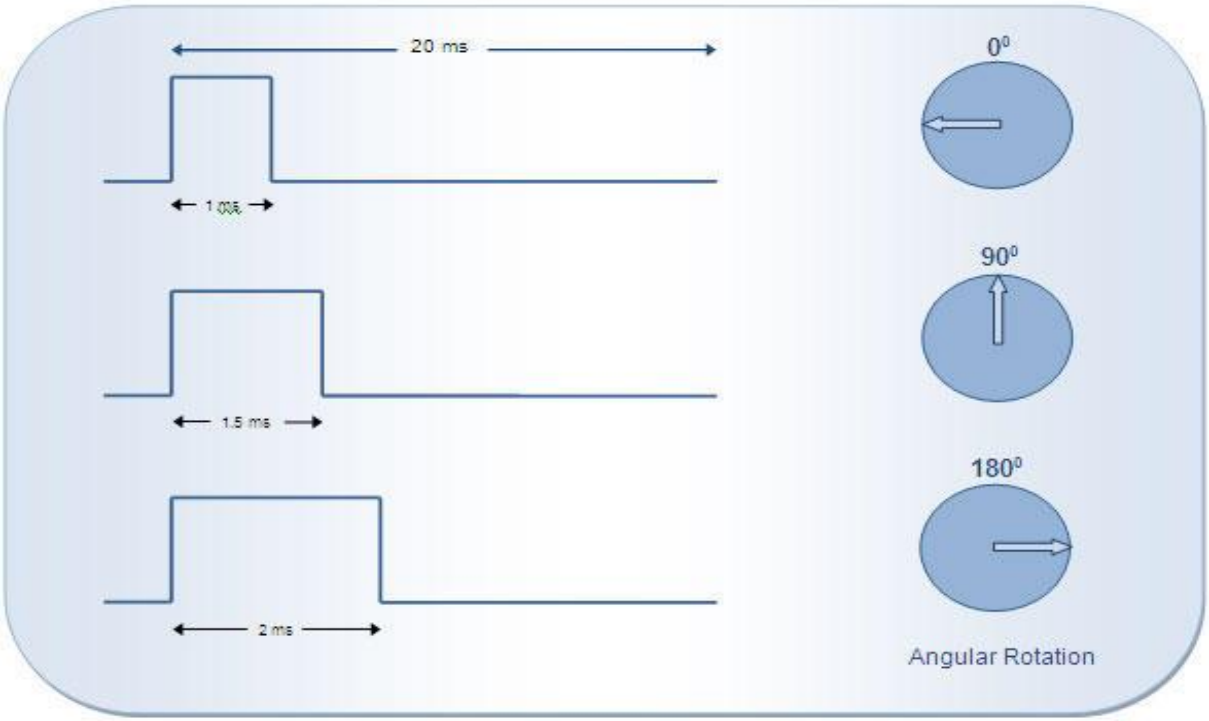

Figura 15 – Controle da posição de um servomotor

A ligação do servomotor pode ser visualida na figura 16.

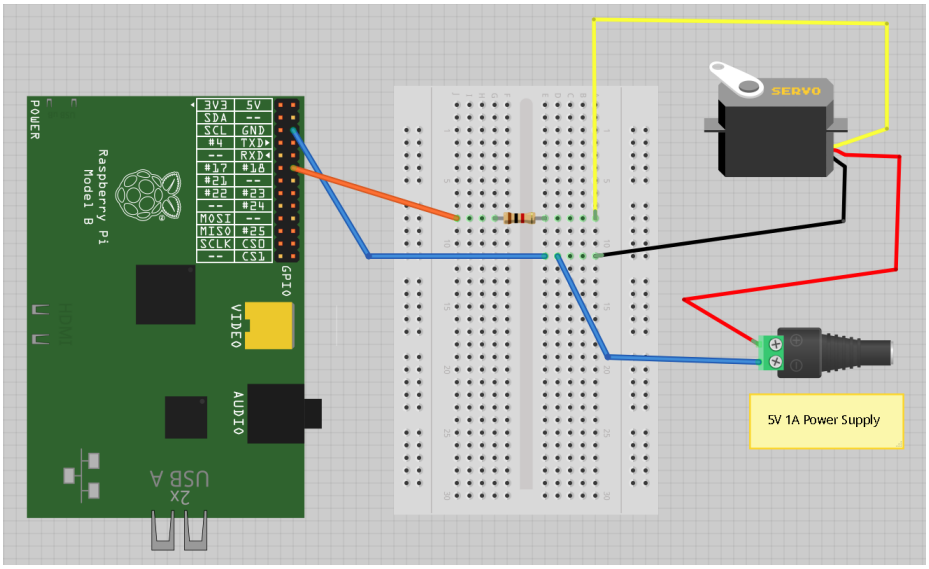

Figura 16 – Conexões do servomotor ao Raspberry Pi

4.4 CONCLUSÃO

A biblioteca BCM2835 disponibiliza funções para criação e controle de PWMs, o que torna o controle do servomotor possível. Esta biblioteca apresentou resultados superiores a biblioteca WiringPi, não apresentando repique na saída. Foi utilizado o GPIO18, representado pelo pino 7 da Raspberry Pi.

REFERÊNCIAS BIBLIOGRÁFICAS

Primeiros passos para configuração da raspberryPI. Disponível em: < http://rpi.nersd.ifsc.edu.br/?p=18>. Acesso em 29 Jun. 2015

How to interface Servo Motor with PIC Microcontroller. Disponível em: < http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-servo-motor-circuit>. Acesso em 29 Jun. 2015

Controlling Servo Motors Disponível em: < http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-servo-motor-circuit>. Acesso em 29 Jun. 2015