## linha horizontal



Controlador de sistema de aquecimento

09/09/2019

**─**

Gabriel de Souza Nogueira da Silva

Gabriel Ribeiro Camelo

Universidade Federal do Ceará

Rua Cel. Estanislau Frota, 563

Sobral, Ceará, 62010-560

# Visão geral

Implementar um software em tempo real na linguagem C para controlar a temperatura de um aquecedor simulado obedecendo os requisitos físicos do sistema e usando como código base o exemplo “controlemanual.c” dado junto com os arquivos do simulador.

# Objetivos

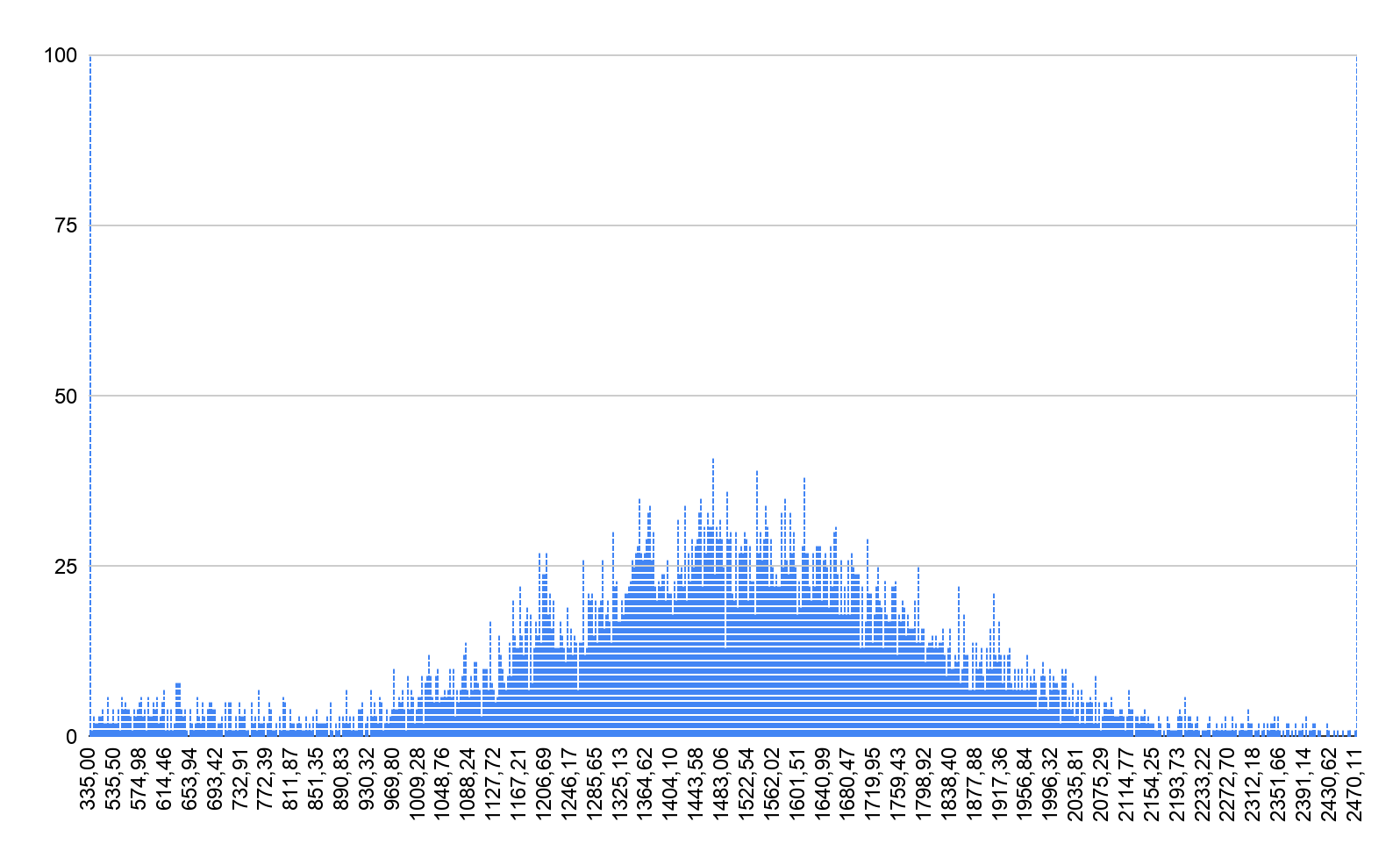
1. Criar um laço de controle como tarefa periódica para a temperatura;
2. Uso das entradas Na e Ni e da saída de água Nf para o controle;
3. Informações na tela sobre a situação corrente.

# Especificações

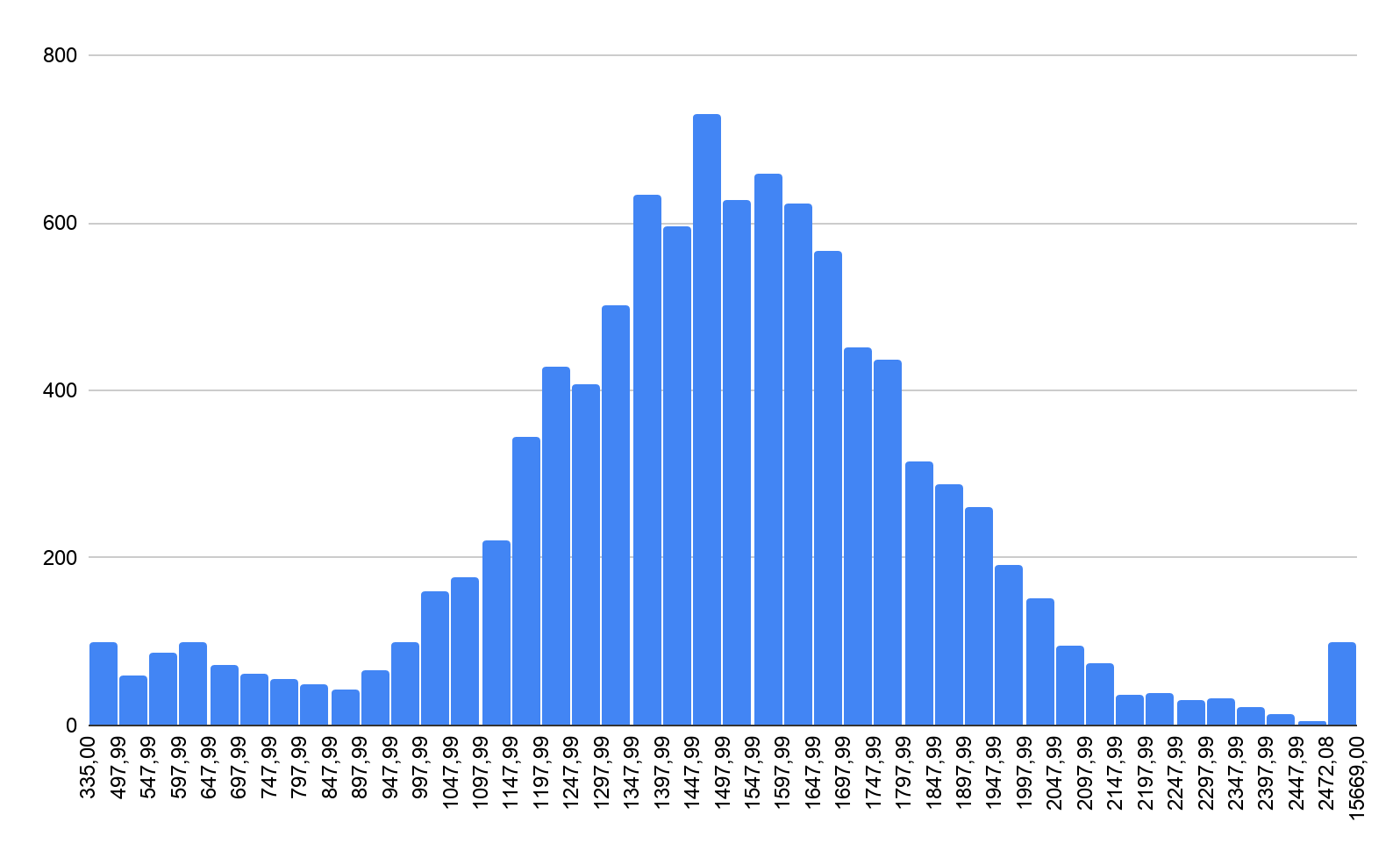
Quando o programa é iniciado é necessário que o usuário digite a temperatura desejada e o software se encarregará de manter a temperatura no valor desejado.

OBS.: Para mais informações sobre o código consultar a seção de anexo (Página 3).

## **Análise gráfica dos tempos de resposta**



No gráfico acima foi utilizado um total de 10000 amostras de tempo de resposta do mesmo programa enquanto ele funcionava controlando a temperatura do aquecedor. Como são 10000 valores a exibição valor por valor tem uma visão prejudicada pois o gráfico fica muito extenso; logo no gráfico abaixo temos um histograma mais simplificado do programa fazendo uma média entre os valores e exibindo para melhor análise.



Portanto, concluímos que o período de 30 ms é mais que suficiente para o funcionamento do sistema visto que o maior tempo foi de 15669 us, 15,6 ms aproximadamente, fazendo com que o deadline da tarefa sempre seja cumprido. Além disso, se ainda quisermos podemos diminuir o deadline para 20 ms e no pior caso ainda estaria cumprindo o deadline estabelecido.

# 

# 

# Anexos

/\*

\* EQUIPE:

\* Gabriel de Souza Nogueira da Silva

\* Gabriel Ribeiro Camelo

\* Execute o programa em uma máquina linux com o seguinte comando:

\* gcc -o trab\_final trab\_final.c -lrt

\*

\*

\*/

#include <unistd.h>

#include <time.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <string.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#include <netdb.h>

#define NSEC\_PER\_SEC (1000000000) // Numero de nanosegundos em um milissegundo

#define N\_AMOSTRAS 10000 // Numero de amostras (medicoes) coletadas

#define FALHA 1

char teclado[1000];

double valor;

char msg\_enviada[1000], msg\_enviada\_clone[1000];

char msg\_recebida[1000];

int nrec;

long atraso\_inicio[N\_AMOSTRAS]; // Medicoes do atraso ateh inicio em microsegundos

long atraso\_fim[N\_AMOSTRAS]; // Medicoes do atraso ateh o fim em microsegundos

int cria\_socket\_local(void)

{

int socket\_local; /\* Socket usado na comunicacao \*/

socket\_local = socket( PF\_INET, SOCK\_DGRAM, 0);

if (socket\_local < 0) {

perror("socket");

return -1;

}

return socket\_local;

}

struct sockaddr\_in cria\_endereco\_destino(char \*destino, int porta\_destino)

{

struct sockaddr\_in servidor; /\* Endereco do servidor incluindo ip e porta \*/

struct hostent \*dest\_internet; /\* Endereco destino em formato proprio \*/

struct in\_addr dest\_ip; /\* Endereco destino em formato ip numerico \*/

if (inet\_aton ( destino, &dest\_ip ))

dest\_internet = gethostbyaddr((char \*)&dest\_ip, sizeof(dest\_ip), AF\_INET);

else

dest\_internet = gethostbyname(destino);

if (dest\_internet == NULL) {

fprintf(stderr,"Endereco de rede invalido\n");

exit(FALHA);

}

memset((char \*) &servidor, 0, sizeof(servidor));

memcpy(&servidor.sin\_addr, dest\_internet->h\_addr\_list[0], sizeof(servidor.sin\_addr));

servidor.sin\_family = AF\_INET;

servidor.sin\_port = htons(porta\_destino);

return servidor;

}

void envia\_mensagem(int socket\_local, struct sockaddr\_in endereco\_destino, char \*mensagem)

{

/\* Envia msg ao servidor \*/

if (sendto(socket\_local, mensagem, strlen(mensagem)+1, 0, (struct sockaddr \*) &endereco\_destino, sizeof(endereco\_destino)) < 0 )

{

perror("sendto");

return;

}

}

int recebe\_mensagem(int socket\_local, char \*buffer, int TAM\_BUFFER)

{

int bytes\_recebidos; /\* Numero de bytes recebidos \*/

/\* Espera pela msg de resposta do servidor \*/

bytes\_recebidos = recvfrom(socket\_local, buffer, TAM\_BUFFER, 0, NULL, 0);

if (bytes\_recebidos < 0)

{

perror("recvfrom");

}

return bytes\_recebidos;

}

int main(int argc, char\* argv[])

{

struct timespec t, t\_inicio, t\_fim;

int amostra = 0; // Amostra corrente

long int periodo = 30000000; // 30ms

if (argc < 3) {

fprintf(stderr,"Uso: controlemanual <endereco> <porta>\n");

fprintf(stderr,"<endereco> eh o endereco IP da caldeira\n");

fprintf(stderr,"<porta> eh o numero da porta UDP da caldeira\n");

fprintf(stderr,"Exemplo de uso:\n");

fprintf(stderr," controlemanual localhost 12345\n");

exit(FALHA);

}

int porta\_destino = atoi( argv[2]);

int socket\_local = cria\_socket\_local();

struct sockaddr\_in endereco\_destino = cria\_endereco\_destino(argv[1], porta\_destino);

// Tarefa periodica iniciará em 1 segundo

//t.tv\_sec++;

int temperatura\_user;

float haltura, temperatura\_sist;

printf("digite o valor da temperatura desejada\n");

scanf("%d", &temperatura\_user);

char\* ler(char\* consulta){//ler as variaveis do simulador

strcpy( msg\_enviada, consulta);

envia\_mensagem(socket\_local, endereco\_destino, msg\_enviada);

nrec = recebe\_mensagem(socket\_local, msg\_recebida, 1000);

msg\_recebida[nrec] = '\0';

return msg\_recebida;

}

void altera(char\* consulta, float valor){//altera os valores das variaveis no simulador

sprintf( msg\_enviada, consulta, valor);

envia\_mensagem(socket\_local, endereco\_destino, msg\_enviada);

nrec = recebe\_mensagem(socket\_local, msg\_recebida, 1000);

msg\_recebida[nrec] = '\0';

}

void altera\_altura(){//equilibra a altura entre 2.6 e 2 m

if(haltura >= 2.6){

altera("anf%lf", 100.0);

altera("ani%lf", 0.0);

altera("ana%lf", 0.0);

}

if(haltura <= 2){

altera("anf%lf", 0);

}

}

// Le a hora atual, coloca em t

clock\_gettime(CLOCK\_MONOTONIC ,&t);

while(amostra < N\_AMOSTRAS) {

// Espera ateh inicio do proximo periodo

clock\_nanosleep(CLOCK\_MONOTONIC, TIMER\_ABSTIME, &t, NULL);

// Realiza seu trabalho

printf("Coletada a amostra %d\n", amostra);

// Le a hora atual, coloca em t\_inicio

clock\_gettime(CLOCK\_MONOTONIC ,&t\_inicio);

// Calcula atraso observado em microsegundos

atraso\_inicio[amostra] = 1000000\*(t\_inicio.tv\_sec - t.tv\_sec) + (t\_inicio.tv\_nsec - t.tv\_nsec)/1000;

char\* msg\_rec = ler("st-0");

temperatura\_sist = atof(&msg\_rec[3]);

printf(">>>%f<<<\n", temperatura\_sist);

if (temperatura\_user > temperatura\_sist){

msg\_rec = ler("sh-0");

haltura = atof(&msg\_rec[3]);

altera\_altura();

altera("ani%lf", 0.0);

altera("ana%lf", 10.0);

}

if(temperatura\_user < temperatura\_sist){

msg\_rec = ler("sh-0");

haltura = atof(&msg\_rec[3]);

altera\_altura();

altera("ana%lf", 0.0);

altera("ani%lf", 10.0);

}

// Le a hora atual, coloca em t\_fim

clock\_gettime(CLOCK\_MONOTONIC ,&t\_fim);

// Calcula o tempo de resposta observado em microsegundos

atraso\_fim[amostra++] = 1000000\*(t\_fim.tv\_sec - t.tv\_sec) + (t\_fim.tv\_nsec - t.tv\_nsec)/1000;

// Calcula inicio do proximo periodo

t.tv\_nsec += periodo;

while (t.tv\_nsec >= NSEC\_PER\_SEC) {

t.tv\_nsec -= NSEC\_PER\_SEC;

t.tv\_sec++;

}

}

printf("Atrasos ateh o inicio da execucao incluem overhead,release jitter, interferencias\n");

printf("Tempo de resposta vai desde a chegada ateh a conclusao\n\n");

printf("Atraso inicio Tempo de resposta\n");

FILE\* dados\_i;

FILE\* dados\_f;

dados\_i = fopen("dados\_inicio.txt", "w+");

if(dados\_i == NULL){

printf("Erro, nao foi possivel abrir o arquivo\n");

exit(1);

}

dados\_f = fopen("dados\_fim.txt", "w+");

if(dados\_f == NULL){

printf("Erro, nao foi possivel abrir o arquivo\n");

exit(1);

}

for( int i=0; i<N\_AMOSTRAS; ++i){

printf("Atraso inicio=%4ldus Tempo de resposta=%4ldus\n", atraso\_inicio[i], atraso\_fim[i]);

fprintf(dados\_i, "%4ld \n", atraso\_inicio[i]);

fprintf(dados\_f, "%4ld \n", atraso\_fim[i]);

}

fclose(dados\_i);

fclose(dados\_f);

}

\* EQUIPE:

\* Gabriel de Souza Nogueira da Silva

\* Gabriel Ribeiro Camelo

\* Execulte o programa em uma maquina linux com o seguinte comando:

\* gcc -o trab\_final trab\_final.c -lrt

\*

\*

\*/

#include <unistd.h>

#include <time.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <string.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#include <netdb.h>

#define NSEC\_PER\_SEC (1000000000) // Numero de nanosegundos em um milissegundo

#define N\_AMOSTRAS 10000 // Numero de amostras (medicoes) coletadas

#define FALHA 1

char teclado[1000];

double valor;

char msg\_enviada[1000], msg\_enviada\_clone[1000];

char msg\_recebida[1000];

int nrec;

long atraso\_inicio[N\_AMOSTRAS]; // Medicoes do atraso ateh inicio em microsegundos

long atraso\_fim[N\_AMOSTRAS]; // Medicoes do atraso ateh o fim em microsegundos

int cria\_socket\_local(void)

{

int socket\_local; /\* Socket usado na comunicacao \*/

socket\_local = socket( PF\_INET, SOCK\_DGRAM, 0);

if (socket\_local < 0) {

perror("socket");

return -1;

}

return socket\_local;

}

struct sockaddr\_in cria\_endereco\_destino(char \*destino, int porta\_destino)

{

struct sockaddr\_in servidor; /\* Endereco do servidor incluindo ip e porta \*/

struct hostent \*dest\_internet; /\* Endereco destino em formato proprio \*/

struct in\_addr dest\_ip; /\* Endereco destino em formato ip numerico \*/

if (inet\_aton ( destino, &dest\_ip ))

dest\_internet = gethostbyaddr((char \*)&dest\_ip, sizeof(dest\_ip), AF\_INET);

else

dest\_internet = gethostbyname(destino);

if (dest\_internet == NULL) {

fprintf(stderr,"Endereco de rede invalido\n");

exit(FALHA);

}

memset((char \*) &servidor, 0, sizeof(servidor));

memcpy(&servidor.sin\_addr, dest\_internet->h\_addr\_list[0], sizeof(servidor.sin\_addr));

servidor.sin\_family = AF\_INET;

servidor.sin\_port = htons(porta\_destino);

return servidor;

}

void envia\_mensagem(int socket\_local, struct sockaddr\_in endereco\_destino, char \*mensagem)

{

/\* Envia msg ao servidor \*/

if (sendto(socket\_local, mensagem, strlen(mensagem)+1, 0, (struct sockaddr \*) &endereco\_destino, sizeof(endereco\_destino)) < 0 )

{

perror("sendto");

return;

}

}

int recebe\_mensagem(int socket\_local, char \*buffer, int TAM\_BUFFER)

{

int bytes\_recebidos; /\* Numero de bytes recebidos \*/

/\* Espera pela msg de resposta do servidor \*/

bytes\_recebidos = recvfrom(socket\_local, buffer, TAM\_BUFFER, 0, NULL, 0);

if (bytes\_recebidos < 0)

{

perror("recvfrom");

}

return bytes\_recebidos;

}

int main(int argc, char\* argv[])

{

struct timespec t, t\_inicio, t\_fim;

int amostra = 0; // Amostra corrente

long int periodo = 30000000; // 30ms

if (argc < 3) {

fprintf(stderr,"Uso: controlemanual <endereco> <porta>\n");

fprintf(stderr,"<endereco> eh o endereco IP da caldeira\n");

fprintf(stderr,"<porta> eh o numero da porta UDP da caldeira\n");

fprintf(stderr,"Exemplo de uso:\n");

fprintf(stderr," controlemanual localhost 12345\n");

exit(FALHA);

}

int porta\_destino = atoi( argv[2]);

int socket\_local = cria\_socket\_local();

struct sockaddr\_in endereco\_destino = cria\_endereco\_destino(argv[1], porta\_destino);

// Tarefa periodica iniciará em 1 segundo

//t.tv\_sec++;

int temperatura\_user;

float haltura, temperatura\_sist;

printf("digite o valor da temperatura desejada\n");

scanf("%d", &temperatura\_user);

char\* ler(char\* consulta){//ler as variaveis do simulador

strcpy( msg\_enviada, consulta);

envia\_mensagem(socket\_local, endereco\_destino, msg\_enviada);

nrec = recebe\_mensagem(socket\_local, msg\_recebida, 1000);

msg\_recebida[nrec] = '\0';

return msg\_recebida;

}

void altera(char\* consulta, float valor){//altera os valores das variaveis no simulador

sprintf( msg\_enviada, consulta, valor);

envia\_mensagem(socket\_local, endereco\_destino, msg\_enviada);

nrec = recebe\_mensagem(socket\_local, msg\_recebida, 1000);

msg\_recebida[nrec] = '\0';

}

void altera\_altura(){//equilibra a altura entre 2.6 e 2 m

if(haltura >= 2.6){

altera("anf%lf", 100.0);

altera("ani%lf", 0.0);

altera("ana%lf", 0.0);

}

if(haltura <= 2){

altera("anf%lf", 0);

}

}

// Le a hora atual, coloca em t

clock\_gettime(CLOCK\_MONOTONIC ,&t);

while(amostra < N\_AMOSTRAS) {

// Espera ateh inicio do proximo periodo

clock\_nanosleep(CLOCK\_MONOTONIC, TIMER\_ABSTIME, &t, NULL);

// Realiza seu trabalho

printf("Coletada a amostra %d\n", amostra);

// Le a hora atual, coloca em t\_inicio

clock\_gettime(CLOCK\_MONOTONIC ,&t\_inicio);

// Calcula atraso observado em microsegundos

atraso\_inicio[amostra] = 1000000\*(t\_inicio.tv\_sec - t.tv\_sec) + (t\_inicio.tv\_nsec - t.tv\_nsec)/1000;

char\* msg\_rec = ler("st-0");

temperatura\_sist = atof(&msg\_rec[3]);

printf(">>>%f<<<\n", temperatura\_sist);

if (temperatura\_user > temperatura\_sist){

msg\_rec = ler("sh-0");

haltura = atof(&msg\_rec[3]);

altera\_altura();

altera("ani%lf", 0.0);

altera("ana%lf", 10.0);

}

if(temperatura\_user < temperatura\_sist){

msg\_rec = ler("sh-0");

haltura = atof(&msg\_rec[3]);

altera\_altura();

altera("ana%lf", 0.0);

altera("ani%lf", 10.0);

}

// Le a hora atual, coloca em t\_fim

clock\_gettime(CLOCK\_MONOTONIC ,&t\_fim);

// Calcula o tempo de resposta observado em microsegundos

atraso\_fim[amostra++] = 1000000\*(t\_fim.tv\_sec - t.tv\_sec) + (t\_fim.tv\_nsec - t.tv\_nsec)/1000;

// Calcula inicio do proximo periodo

t.tv\_nsec += periodo;

while (t.tv\_nsec >= NSEC\_PER\_SEC) {

t.tv\_nsec -= NSEC\_PER\_SEC;

t.tv\_sec++;

}

}

printf("Atrasos ateh o inicio da execucao incluem overhead,release jitter, interferencias\n");

printf("Tempo de resposta vai desde a chegada ateh a conclusao\n\n");

printf("Atraso inicio Tempo de resposta\n");

FILE\* dados\_i;

FILE\* dados\_f;

dados\_i = fopen("dados\_inicio.txt", "w+");

if(dados\_i == NULL){

printf("Erro, nao foi possivel abrir o arquivo\n");

exit(1);

}

dados\_f = fopen("dados\_fim.txt", "w+");

if(dados\_f == NULL){

printf("Erro, nao foi possivel abrir o arquivo\n");

exit(1);

}

for( int i=0; i<N\_AMOSTRAS; ++i){

printf("Atraso inicio=%4ldus Tempo de resposta=%4ldus\n", atraso\_inicio[i], atraso\_fim[i]);

fprintf(dados\_i, "%4ld \n", atraso\_inicio[i]);

fprintf(dados\_f, "%4ld \n", atraso\_fim[i]);

}

fclose(dados\_i);

fclose(dados\_f);

}