TRAVALHO INDIVIDUAL # 1
COMUNICAÇÃO UART

Vilmey Francisco Romano Filho - 11/0021380

Programa de Graduação em Engenharia Eletrônica, Faculdade Gama
Universidade de Brasília
Gama, DF, Brasil
email: vilmeyr@gmail.com

1. OBJETIVO

Esta trabalho prático tem a finalidade exemplificar e reunir técnicas aprendidas em sala de aula para a solução de problemas. Nesta prática damos ênfase a comunicação entre um sistema complexo com SO embarcado e um microcontrolado, neste caso um Raspberry PI e um arduino (via UART).

2. INTRODUÇÃO

O microcontrolador ATmega possui muitas ferramentas úteis ao engenheiro, este controlador pode se communicar com outros dispositivos atravéz de protocolo de comunicação serial UART(Universal asynchronous receiver/transmitter). Para a comunicação entre a (Raspberry Pi) e o microcontrolator esse protocolo se mostra eficiente, pois ambos não tem que compartilhar um sinal de clock.
O programa em C deve ser capaz de receber comandos via terminal (Raspberry PI) e ser capaz de enviar e receber strings, inteiros e ponto flutuantes do controlador.

3. ESPECIFICAÇÃO

3.1. Escrita de dados

Todas as mensagens de solicitação de dados enviadas pela porta serial devem seguir o padrão: Código do comando solicitado figura[1] + Quatro últimos dígitos da matrícula em formado char. Exemplo de Mensagem com o comando 0xA3 (Hexadecimal) = 163 (Decimal) e a matrícula ?1380? ao final: char[] = 163, 1, 3, 8, 0; Já as mensagens de envio de dados deverão ser compostas por: Comando + Dado + Matrícula. No caso do envio de uma String, o formato deverá ser: Comando + Tamanho da String (1 byte) + String + Matrícula.

3.2. Leitura de dados

A leitura de dados deve seguir o padrão de retorno da Tabela 1. Para valores inteiros (int) ou reais (float), deverão ser lidos 4 bytes e armazenados em uma variável int ou float, respectivamente. Para leitura de strings, a mensagem de retorno irá conter o número total de caracteres da string no primeiro byte (Valor entre 0 e 255), em seguida, é possível ler o conteúdo da mensagem que deverá ser armazenada em um array de char (char[]);

Tabela 1 - Códigos do Protocolo de Comunicação - Solicitação de Informações

Código	Comando de Solicitação de Dados	Mensagem de Retorno
0xA1	Solicitação de dado inteiro: ***integer***	int (4 bytes)
0xA2	Solicitação de dado real: ***float***	float (4 bytes)
0xA3	Solicitação de dado do tipo string: ***char[]***	char (1 byte com o tamanho da string) + char[] (nbytes com o conteúdo da string)

Tabela 2 - Códigos do Protocolo de Comunicação - Envio de Dados

Código	Comando de Envio de Dados	Mensagem de Retorno
0xB1	Envio de um dado no formato ***integer***	int (4 bytes)
0xB2	Envio de um dado no formato ***float***	float (4 bytes)
0xB3	Envio de uma string: ***char[]***	char (1 byte com o tamanho da string) + char[] (nbytes com o conteúdo da string)

Fig. 1. Comandos para envio e tipo de retorno.

4. IMPLEMENTAÇÃO

Foi criado um menu com as opções de solicitar: inteiro, float ou string, assim como enviar: inteiro, float ou string ao controlador. Quando o usuário seliciona a opção desejada é direcionado a uma função específica que é responsável por tratar os dados e realizar a comunicação devidamente.

A main do programa passa os parâmetros basicos para os procedimentos como o comando, a matricula e o filestream. Já dentro dos procedimentos a sua função irá variar de acordo com o tipo de variável (de envio ou recebimento).

O que ocorre no geral é a concatenação das informações e consequentemente o envio deste ao arduino por meio do

comando *write*.

As concatenaçoes foram feitas utilizando o comando *memcpy*, que é uma versão mais baixo nível em relação as funções formatadas.

Quando é necessário a leitura de um retorno utilizamos o comando *read* que irá armazenar a resultado em um *void buffer*.

Os dados recebidos são apresentados em tela com o comando *printf*.

5. CONCLUSÕES

A prova prática foi contrutiva no tocante a parte de comunicação entre dispositivos, pois a mesma englobou um protocolo bastante usado para comunicação entre dispositivos e perifericos.

O experimeto ocorreu bem, apesar das dificuldades inicias para transmitir os comando para o controlador. Porém uma vez aprendido esses conceitos foi possivel extende-los ao esperado no final do trabalho prático.

6. CÓDIGO

```c
#include <termios.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

#define DEVICE_FILE    "/dev/ttyAMA0"


//Configura os parametros da comunicacao UART
void Uart_Config(int * uart0_filestream)
{
  struct termios options;
  tcgetattr(*uart0_filestream, &options);
  options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
     // Set baud rate
  options.c_iflag = IGNPAR;
  options.c_oflag = 0;
  options.c_lflag = 0;
  tcflush(*uart0_filestream, TCIFLUSH);
  tcsetattr(*uart0_filestream, TCSANOW, &options)
    ;
}

//Constroi o menu de opcoes
void Menu ()
{
  printf("(1) Solicita int\n");
  printf("(2) Solicita float\n");
  printf("(3) Solicita char\n");
  printf("(4) Envia int\n");
  printf("(5) Envia float\n");
  printf("(6) Envia char\n");

  printf("Digite uma opÃ§Ã£o: ");
}


void Solicita_int (char Comando, char *matricula,
     int *uart0_filestream)
{
  int retorno;
  char chave[60] = {Comando};

  memcpy(&chave[1], matricula, 4);

  *uart0_filestream = open(DEVICE_FILE, O_RDWR |
    O_NOCTTY |O_NDELAY);

  int tx_length = write(*uart0_filestream, chave,
     (int)strlen(chave));
    if (tx_length < 0)
      printf("Erro na transmissao - UART TX\n");


  int rx_length = read(*uart0_filestream, &
    retorno, sizeof(int));
    if (rx_length < 0)
      perror("Falha na leitura");

    else if (rx_length == 0)
      printf("Nenhum dado disponivel\n");

    else
      printf("\n|%i bytes lidos|", rx_length);

  close(*uart0_filestream);

  printf("\n\nRetorno int: %d\n\n\n", retorno);
}

void Solicita_float (char Comando, char *
    matricula, int *uart0_filestream)
{
  float retorno;
  char chave[60] = {Comando};

  memcpy(&chave[1], matricula, 4);

  *uart0_filestream = open(DEVICE_FILE, O_RDWR |
    O_NOCTTY |O_NDELAY);

  int tx_length = write(*uart0_filestream, chave,
     (int)strlen(chave));
    if (tx_length < 0)
      printf("Erro na transmissao - UART TX\n");

  int rx_length = read(*uart0_filestream, &
    retorno, sizeof(float));
    if (rx_length < 0)
      perror("Falha na leitura");

    else if (rx_length == 0)
      printf("Nenhum dado disponivel\n");

    else
      printf("\n|%i bytes lidos|", rx_length);


  close(*uart0_filestream);

  printf("\n\nRetorno float: %f\n\n\n", retorno);
}

void Solicita_char (char Comando, char *matricula
    , int *uart0_filestream)
{
  char retorno[256], string[256];
  char chave[60] = {Comando};

  memcpy(&chave[1], matricula, 4);

  *uart0_filestream = open(DEVICE_FILE, O_RDWR |
    O_NOCTTY |O_NDELAY);

  int tx_length = write(*uart0_filestream, chave,
     (int)strlen(chave));
    if (tx_length < 0)
      printf("Erro na transmissao - UART TX\n");

  int rx_length = read(*uart0_filestream, retorno
    , 255);
    if (rx_length < 0)
      perror("Falha na leitura");

    else if (rx_length == 0)
      printf("Nenhum dado disponivel\n");

    else
      printf("| %i bytes lidos |", rx_length);

  close(*uart0_filestream);
```

```
119
   printf("\n\nTamanho recebido : %d \n", (int)
     retorno[0]); //Imprime o tamanho da string
     recebida
121   printf("Retorno string: (%s) (%s)\n\n\n",
     string, retorno);
}
123
void Envia_int (char Comando, char *matricula,
    int *uart0_filestream)
125 {
   int i, retorno;
127   char chave[60] = {Comando};

129   printf("\nDigite o inteiro a ser enviado: ");
   scanf("%d", &i);
131
   memcpy(&chave[1], &i, 4);
133   memcpy(&chave[5], matricula, 4);


135
   *uart0_filestream = open(DEVICE_FILE, O_RDWR |
     O_NOCTTY |O_NDELAY);
137
   int tx_length = write(*uart0_filestream, chave,
      (int)strlen(chave));
139     if (tx_length < 0)
       printf("Erro na transmissao - UART TX\n");
141
   int rx_length = read(*uart0_filestream, &
     retorno, sizeof(int));
143   if (rx_length < 0)
     perror("Falha na leitura");
145
   else if (rx_length == 0)
147     printf("Nenhum dado disponivel \n");

149   else
     printf("\n|%i bytes lidos |", rx_length);
151


153   close(*uart0_filestream);

155   printf("\nRetorno int: %d\n\n\n", retorno);
}
157
void Envia_float (char Comando, char *matricula,
    int *uart0_filestream)
159 {
   float i, retorno;
161   char chave[60] = {Comando};

163   printf("\nDigite o real a ser enviado: ");
   scanf("%f", &i);
165
   memcpy(&chave[1], &i, 4);
167   memcpy(&chave[5], matricula, 4);


169
   *uart0_filestream = open(DEVICE_FILE, O_RDWR |
     O_NOCTTY |O_NDELAY);
171
   int tx_length = write(*uart0_filestream, chave,
      (int)strlen(chave));
173     if (tx_length < 0)
       printf("Erro na transmissao - UART TX\n");
175
   int rx_length = read(*uart0_filestream, &
     retorno, sizeof(float));
177     if (rx_length < 0)
       perror("Falha na leitura \n");
179
     else if (rx_length == 0)
181       printf("Nenhum dado disponivel \n");

183     else
       printf("\n|%i bytes lidos |", rx_length);
185
   close(*uart0_filestream);
187
   printf("\nRetorno float: %f\n\n\n", retorno);
189 }


191
void Envia_char (char Comando, char *matricula,
    int *uart0_filestream)
193 {
   char retorno[256];
195   int tam;
   char tamanho_msg, buffer[60], chave[60] = {
     Comando};
197
   printf("\nDigite a string a ser enviada: ");
199
   getchar();
201   fgets(buffer, 60, stdin); //String a ser
     enviada
     tamanho_msg = (char)strlen(buffer);
203
     buffer[strcspn(buffer, "\n")] = 0;
205
   memcpy(&chave[1], &tamanho_msg, 1);
207   memcpy(&chave[2], buffer, (int)strlen(buffer));
   memcpy(&chave[2+((int)strlen(buffer))],
     matricula, 4);
209


211   *uart0_filestream = open(DEVICE_FILE, O_RDWR |
     O_NOCTTY |O_NDELAY);
   int tx_length = write(*uart0_filestream, chave,
      (int)strlen(chave));
213     if (tx_length < 0)
       printf("Erro na transmissao - UART TX\n");
215
   int rx_length = read(*uart0_filestream, retorno
     , 1);
217     if (rx_length < 0)
       perror("Falha na leitura");
219
     else if (rx_length == 0)
221       printf("Nenhum dado disponivel \n");

223     else
       printf("\n|%i bytes lidos |", rx_length);
225
   close(*uart0_filestream);
227
   printf("\nRetorno string: %s\n\n\n", retorno);
229 }

231 int main ()
{
233   int uart0_filestream = -1;
   char matricula[4] = {'1', '3', '8', '0'};
```

```
235     char RCV_INTEGER   = 0xA1;
        char RCV_FLOAT     = 0xA2;
237     char RCV_CHAR      = 0xA3;
        char SND_INTEGER   = 0xB1;
239     char SND_FLOAT     = 0xB2;
        char SND_CHAR    = 0xB3;
241
        Uart_Config(&uart0_filestream);
243
245     while(1)
        {
247       int opcao;

249       Menu();

251       scanf("%d", &opcao);
          system("clear");
253
          switch(opcao)
255       {
            case 1:
257           Solicita_int(RCV_INTEGER, matricula, &
          uart0_filestream);
              break;
259         case 2:
              Solicita_float(RCV_FLOAT, matricula, &
          uart0_filestream);
261           break;
            case 3:
263           Solicita_char(RCV_CHAR, matricula, &
          uart0_filestream);
              break;
265         case 4:
              Envia_int(SND_INTEGER, matricula, &
          uart0_filestream);
267           break;
            case 5:
269           Envia_float(SND_FLOAT, matricula, &
          uart0_filestream);
              break;
271         case 6:
              Envia_char(SND_CHAR, matricula, &
          uart0_filestream);
273           break;

275       default:
            printf("OpÃ§Ã£o InvÃ¡lida !\n");
277     }
        }
279
        return 0;
281 }
```

Codigo/From_scratch.c