Universidade Estadual de Campinas - UNICAMP

Faculdade de Engenharia Elétrica e Computação - FEEC

Projeto Final EA871 - 2s2015 - Turmas S e E

Professores : Roberto A. Lotufo
Antônio Augusto Fasolo Quevedo

Grupo : Gustavo G. Plensack - RA 155662 - Turma S
Marcelo H. Gumiero - RA 156529 - Turma E

Índice :

● Introdução ao projeto de comunicação entre as placas :

Durante o decorrer do semestre, os alunos foram expostos a diversos módulo de aprendizagem. Desde o reconhecimento de funções básicas da placa, até o tratamento e configurações de interrupções de diversas formas. Para a conclusão do semestre letivo dessa disciplina foi escolhido o projeto recomendado pelos docentes: a comunicação entre duas placas.

Dessa forma, foram utilizadas funções e módulos já desenvolvidos durante o semestre e foram criadas novas rotinas e utilizados novos recursos oferecidos pela placa.

Os recursos e funções desenvolvidas a fim de possibilitar a comunicação entre as placas serão, neste relatório, descritos de forma a esclarecer o raciocínio dos alunos frente às dificuldades impostas pelo desenvolvimento do projeto.

● Objetivos:

1. **Inspirações :**

Nosso sistema de chat foi inspirado nos sistemas que ganham cada vez mais popularidade no mercado atualmente, como o WhatsApp e Facebook. Esses dois sistemas tem tamanha aceitação pela simplicidade de uso e eficiência na transmissão de mensagens.

2. **Interface Homem - Máquina Simples :**

Da observação das nossas inspirações, prezamos por desenvolver um sistema que se comunique de forma simples e lógica com o usuário, de modo a torná-lo acessível a qualquer pessoa. Para isso, usamos de sinais visuais e hardware para a autorização da transmissão e leitura das mensagens.

● Funcionamento do sistema:

1. **O conceito da transmissão e recepção de dados Serial e seu uso na transmissão de caracteres :**

A ideia de comunicação serial remete ao envio da mensagem bit a bit, e a verificação da validade da mesma usamos bits de paridade, start e stop bits. Caso essa validade não seja verificada, a mensagem é descartada. Usando dessa forma de comunicação implementamos o seguinte sistema :

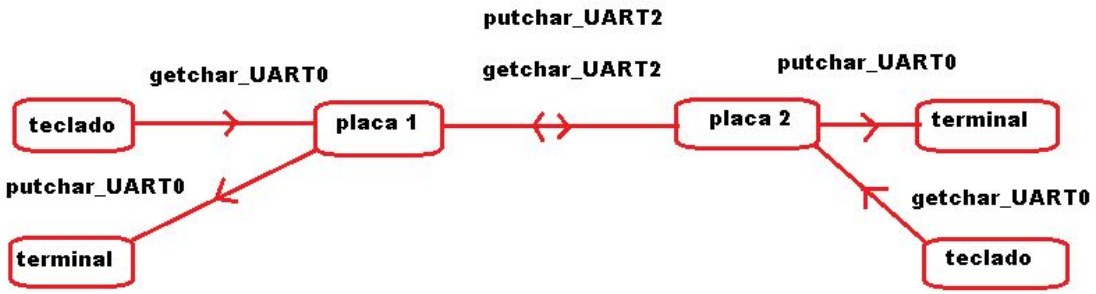

Figura 1: sistema de transmissão entre os componentes

Note que na figura 1, a transmissão é mostrada em duas vias. Ou seja, a sequência "*teclado->placa->placa->terminal"* foi aplicado para as duas placas.

2. Aquisição de dados e armazenagem :

Do modo como implementamos o projeto cada caractere é inicialmente transmitida através da UART0 do PC1 para o MCU1, e em seguida através da UART2 do MCU1 para outro MCU2, onde é armazenada numa string de tamanho definido e aguarda a autorização do MCU1 para ser enviada através da UART0 do MCU2 ao PC2.

3. A Interface Homem - Máquina :

Para dar ao usuário o poder de definir quando será autorizada ou não a leitura da mensagem enviada, ou sua destruição, criamos um sistema de envio de caracteres especiais e flags que permitem tais funções. Para dar o poder ao usuário colocamos nos push buttons essa função, tornando assim o uso simples e intuitivo do sistema. Para a notificação visual de que a mensagem foi recebida e aguarda para ser lida, usamos dos leds da placa shield. Tais recursos são mostrados na figura 2. Mostramos também no display LCD o estado em que se encontra a conexão entre os microcontroladores, existindo os estados de "OK!!" e "FAIL".

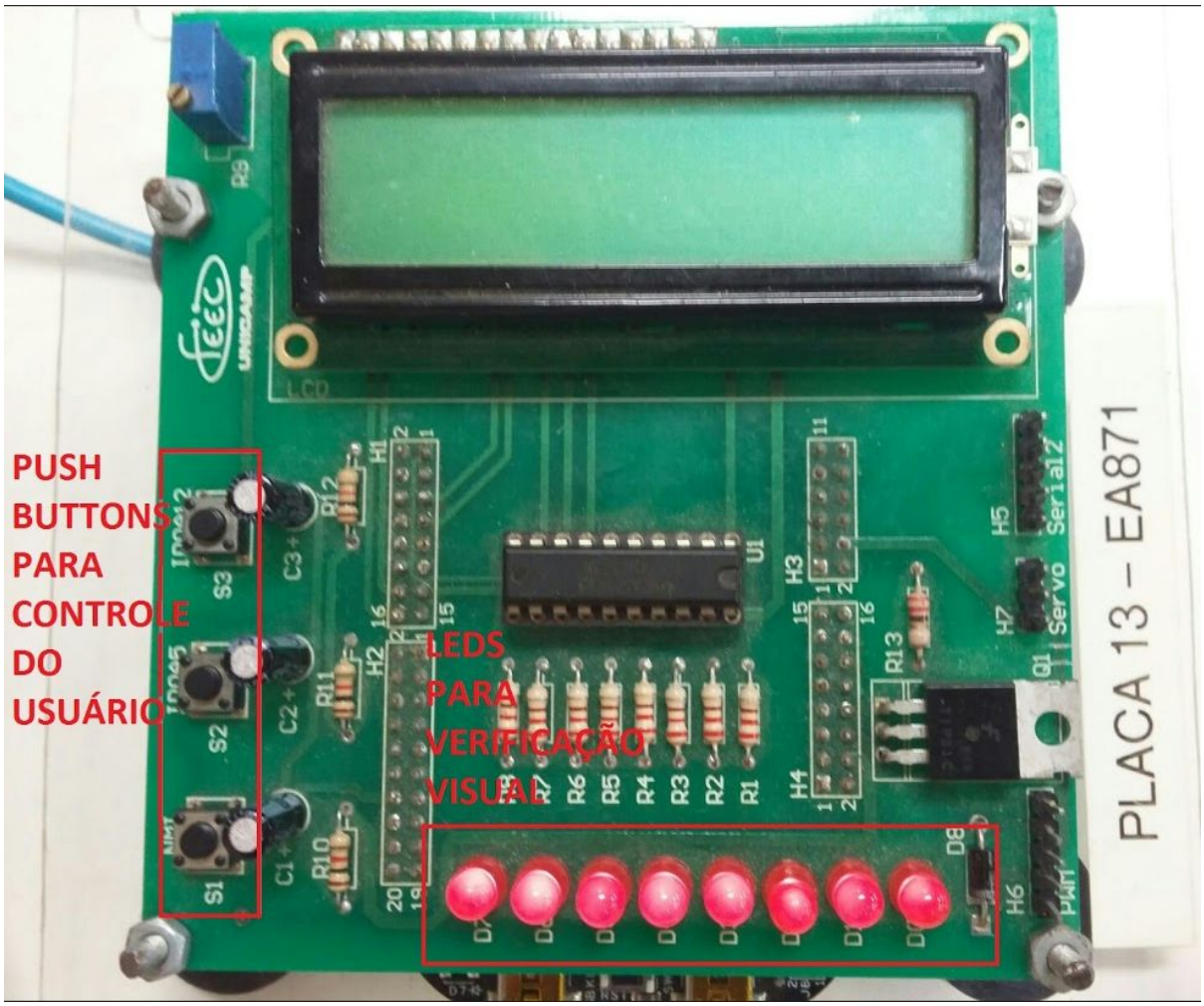

Figura 2 - *push buttons* e *LEDS* utilizados na execução do projeto

● Desenvolvimento :

1. **UART2 - comunicação entre as duas placas e definições :**

Para a sistematização da comunicação pela UART2, usamos uma implentação muito similar a da UART0 feita no programa 10. Para isso usamos as seguintes funções :

void init_UART2(): nela definimos a comunicação com baud rate de 19200, 1 stop bit e 1 start bit, assim como a interrupção via NVIC. Note que diferentemente da UART0, as UART1 e UART2 não necessitam de inicialização de clock, pois pela construção essas usam o mesmo da placa.

```
void init_UART2(){
    SIM_SCGC5 = SIM_SCGC5 | (1<<13); //HABILITA port
    SIM_SCGC4 = SIM_SCGC4 | (1<<12); //HABILITA UART2
    //set mux(RX,TX)
    PORTE_PCR22 = 0x400;
    PORTE_PCR23 = 0x400;
    //rate
    UART2_BDH = 0x00;
    UART2_BDL = 0x44;
    //recebimento, envio de dados
    UART2_C2 = 0x0C;
    UART2_C2 = UART0_C2 | (1<<5);
    //habilita interrupção da UART0
    NVIC_ISER = NVIC_ISER | (1<<14);


    // habilita a interrupacao de TX
    UART0_C2 = UART0_C2 | (1<<7);
}
```

Figura 3 - Função void init_UART2()

void putchar_UART2 (char c) : Função que faz com que os caracteres recebidos sejam colocados no buffer circular para poderem ser enviados na função de UART2_IRQHandler.

```
void putchar_UART2(char c){

    while(n_buffer_2==NMAX){};
    UART2_C2 = 0x2C; // desabilita a interrupcao
    buffer_2[pi_2]=c;
    pi_2 = (pi_2 + 1)%NMAX;
    n_buffer_2++;
    UART2_C2 = 0xAC;  // reabilita a interrupcao
}
```

Figura 4 - void putchar_UART2 (char c)

char getchar_UART2() : realiza a aquisição do char que foi transmitido pela UART2 e obtido na UART2_IRQHandler e o retorna para que possa ser usado de forma satisfatória, caso não haja nenhum dado novo retorna o char 0.

```
char getchar_UART2(){ // retorna caractere transmitido pela UART2, caso o caractere não esteja pronto, retorna 0
    char c;
    if (RX_full_2){ // verifica se existe um novo caractere na UART2 (Register Full)
        c = RX_data_2; // lê o caractere
        RX_full_2=0;
        return c;
    }
    else return 0;       // não existe caractere na UART
}
```

Figura 5 - char getchar_UART2()

void UART2_IRQHandler (void) : Essa é a função responsável por tratar da interrupção NVIC associada a UART2. Nela realizamos o tratamento das interrupções de transmissão e de recepção de dados. Ela é a função responsável pelo envio e recebimento de dados de forma direta, pois ela que escreve e lê no registrador UART2_D, onde os dados são armazenados. Ela mesma desabilita a interrupção de transmissão caso o buffer circular esteja vazio

```
void UART2_IRQHandler(void){
    if(UART2_S1 & UART_S1_RDRF_MASK){ // testa se houve Recepção
        RX_data_2 = UART2_D;
        RX_full_2 = 1;
    }
    else if ((UART2_S1 & UART_S1_TDRE_MASK)){ // testa se houve Transmissão
        UART2_D = buffer_2[po_2]; // envia o caractere pelo registrador de dados da UART2
        po_2=(po_2+1)%NMAX;
        n_buffer_2--;
        if (n_buffer_2==0){
            UART2_C2 = 0x2C; // desabilita a interrupcao de transmissao
        }
    }

}
```

Figura 6 - void UART2_IRQHandler (void)

A ligação entre as placas usando a UART2 foi feita da seguinte forma :

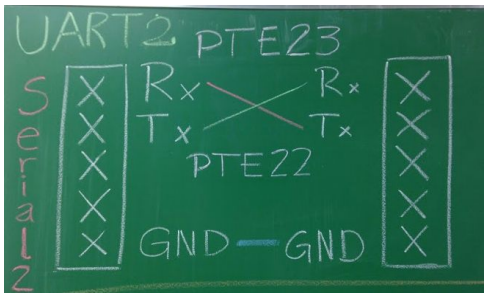

Figura 7 - Esquemático ligação entre placas

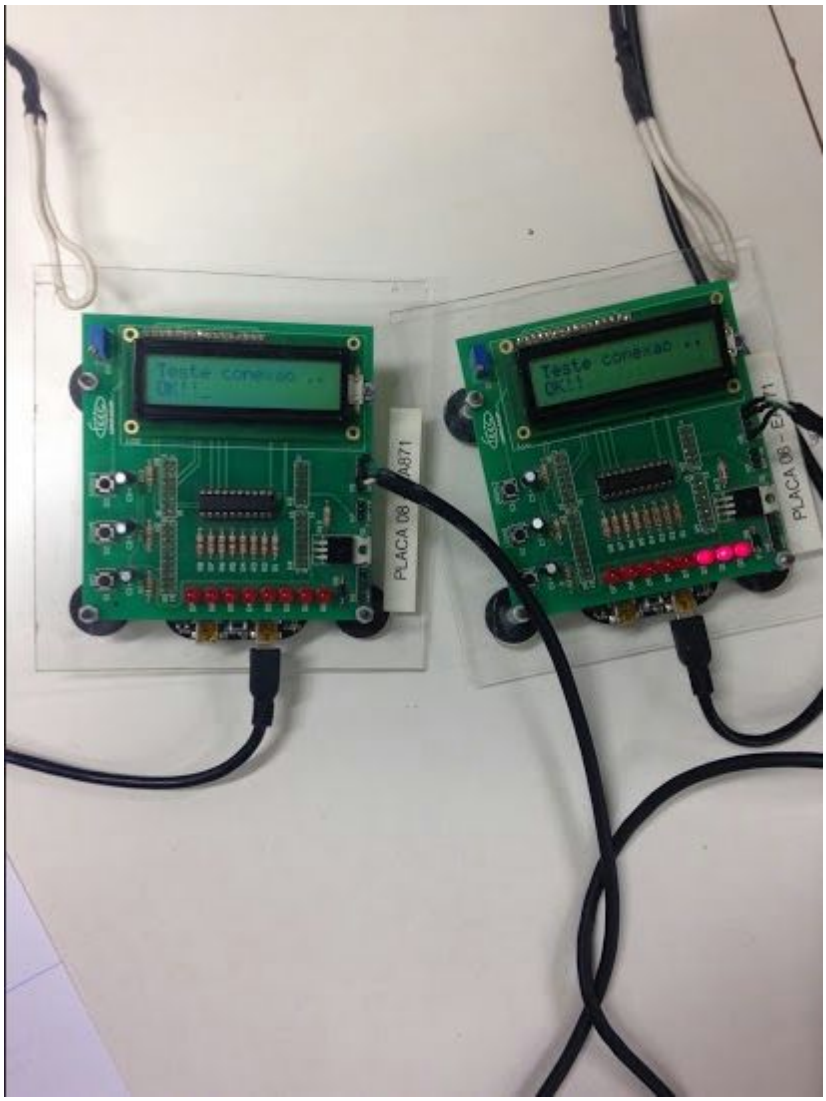

Figura 8 - ligação entre as placas

2. Aquisição de dados e armazenagem :

Nossa aquisição de dados se inicia com o uso da UART0 como forma de receber os caracteres do usuário do PC e na UART2 como comunicação entre as placas. A mensagem é enviada caractere a caractere pela UART2, e é colocada numa string de tamanho definido (*message_2* [TX_MAX + 1]) na placa onde ela foi recebida. O trecho de código a seguir mostra essa implementação.

```
d = getchar_UART2();

caractere_especial = 0;

if(d){

    if(d=='§'){
        autoriza = 1;
        //acende led
        GPIOC_PDOR = 0xFF;
        GPIOC_PSOR = (1<<10); // pulsa LE (Latch Enable) do 74573, em 1 (set)
        GPIOC_PCOR = (1<<10); // em 0 (clear)
        caractere_especial = 1;
    }

    if (d == 254) {
        i = 0 ;
        j = 0 ;
        preenche_espaco(message_2,i); // destroi a mensagem
        caractere_especial = 1;
    }

       if (d == 223){
        sendcmd_LCD(0xc0,4);
        print_LCD("OK!!");
        verifica_conex = 1;
        caractere_especial = 1;
        timer_conex=0;
    }


    if (!caractere_especial) {
        message_2[i] = d; // recebe o dado de mensagem
        i++;
    }
}
```

Figura 9 - Aquisição de dados e armazenagem

3. **Uso do Hardware disponível para desenvolvimento da Interface Homem Máquina :**

Para a implementação da interface homem máquina que se adequasse aos nossos objetivos do projeto, usamos da função int getkey(), que já havia sido desenvolvida em laboratórios anteriores, para dar ao usuário acesso aos comandos de envio de mensagem, destruição da mensagem e leitura da mensagem. Veja a implementação a seguir :

```
switch(getkey()){

case 1: // autoriza leitura
    if(autoriza){
        preenche_espaco(message_2,i);

        if(i){
            print_UART0(message_2);
            i = 0;
            quebralinha();
            j=0;
        }
        autoriza = 0;
        //apaga leds
        GPIOC_PDOR = 0x00;
        GPIOC_PSOR = (1<<10); // pulsa LE (Latch Enable) do 74573, em 1 (set)
        GPIOC_PCOR = (1<<10); // em 0 (clear)
    }
    break;
case 2:
    if(j){
        putchar_UART2('§'); // autoriza a leitura
    }
    break;
case 3:
    putchar_UART2(254); // autoriza a destruicao da mensagem
    break;
}
```

Figura 10 - Uso do getkey para dar acesso aos botões e suas funções

```
d = getchar_UART2();

caractere_especial = 0;

if(d){

    if (d == 223){
        sendcmd_LCD(0xc0,4);
        print_LCD("OK!!");
        verifica_conex = 1;
        caractere_especial = 1;
        timer_conex=0;
    }

    if(d=='§'){
        autoriza = 1;
        //acende led
        GPIOC_PDOR = 0xFF;
        GPIOC_PSOR = (1<<10); // pulsa LE (Latch Enable) do 74573, em 1 (set)
        GPIOC_PCOR = (1<<10); // em 0 (clear)
        caractere_especial = 1;
    }

    if (d == 254) {
        i = 0 ;
        j = 0 ;
        preenche_espaco(message_2,i); // destroi a mensagem
        caractere_especial = 1;
    }

    if (!caractere_especial) {
        message_2[i] = d; // recebe o dado de mensagem
        i++;
    }
}
```

Figura 11 - Tratamento dos envios dos botões 2 e 3 na main

Os caracteres especias escolhidos foram selecionados pois não são dados que frequentemente são enviados numa mensagem em português.

Quando a mensagem está pronta para ser lida notificamos isso ao acender os leds da placa shield, e após sua leitura os apagamos.

O LCD mostra o estado em que se encontra a conexão entre as placas. Seu conteúdo não pode ser mudado diretamente pelo usuário, tem apenas caráter informativo.

4. Tratamento da queda de conexão :

Para o tratamento da queda de conexão, usamos o seguinte sistema :

Criamos uma variável contadora na função principal do programa e a incrementamos a cada ciclo do programa. Dessa forma, quando tal variável chega a um número definido CICLOS_TESTE, enviamos um caractere especial para a UART2. No início do loop, é feito um tratamento desse caractere a fim de verificar a conexão das placas. Caso a variável de verificação estivesse em nível baixo, outro

contador era acionado e, depois de contar CICLOS_TESTE vezes, uma rotina de impressão era permitida através de um "*if*". Isso garante que a conexão só seja dada como desconectada, se realmente os cabos fosse desconectados por um tempo razoável. Colocamos esse tempo como 10% do ciclo de envio de um novo caractere de verificação (Adotado arbitrariamente como o ASCII 223).

Como teste de verificação visual, se a conexão estiver estabelecida, é printado "OK!!" no display LCD, caso contrário é printado "FAIL".

```
        if (d == 223){
         sendcmd_LCD(0xc0,4);
         print_LCD("OK!!");
         verifica_conex = 1;
         caractere_especial = 1;
         timer_conex=0;
      }


      if (!caractere_especial) {
         message_2[i] = d; // recebe o dado de mensagem
         i++;
      }
  }

  if (!verifica_conex) {
      timer_conex++;
      if (timer_conex == CICLOS_TESTE){
         sendcmd_LCD(0xc0,4);
         print_LCD("FAIL");
         timer_conex = 0;
      }
  }
  teste_conexao++ ; // atualiza a variavel pra enviar o teste de conexao

  if (teste_conexao == CICLOS_TESTE ) {
      putchar_UART2 (223) ; // envia o ASCII 223
      teste_conexao = 0;
      sendcmd_LCD(0x80,4);
      print_LCD("Teste conexao ...");
      verifica_conex = 0 ;
  }
```

Figura 12 - Código que implementa o sistema de verificação de conexão

5. Função auxiliar - void preenche_espaco (char *string , int i) :

Para auxiliar no desenvolvimento do projeto, criamos a função void preenche_espaco (char * string , int i), usamos essa função para limpar lixos de memória de mensagens anteriores e manter a formatação da impressão da mensagem.

```
void preenche_espaco (char * string ,int i){

    while (i < TX_MAX){
        string [i] = ' ';
        i++;
    }
    // coloca espacos em branco no fim da mensagem pra manter a formatacao e apagar lixos de mensagens anteriores
}
```

Figura 13 - função auxiliar para a impressão de strings

● Dificuldades encontradas :

1. **Armazenagem de dados :**

Como o objetivo do projeto era criar um módulo de transmissão semelhante aos encontrados no mercado atual, nos quais a transmissão acontece por mensagens completas e não por caracteres separados, tivemos que pensar num modo de transmissão de strings. Entretanto, devido à limitação do registrador de dados da UART2, não foi possível implementar a transmissão direta da mensagem do modo como implementamos nossas funções de getchar_UART2 e putchar_UART2. Dessa forma, como já fora mencionado em “Aquisição de dados e armazenagem”, optamos por transmitir caracteres separados e armazená-los em uma string, chamada message_2, que seria preenchida somente no sistema que recebia as informações. Após preenchida, definimos a um modo de autorização (mostrada em Desenvolvimento.3.) fornecido pelo usuário de transmissão. Ou seja, ao pressionar as teclas no computador 1, as informações eram transmitidas e armazenadas no computador 2, sendo que este dependia de uma autorização fornecida pelo computador 1 para que tais informações fossem impressas no terminal.

2. **Sistema de flags para fazer a verificação da transmissão implementação da interface Homem - Máquina de forma inteligente e lógica :**

O sistema de flags, utilizado para a autorização da impressão de mensagem e dá destruição de mensagens, gerou dúvidas a respeito de como iríamos comunicar as duas placas através do *push button*. Para solucionar tal problema foi utilizado um sistema de transmissão que mudava as flags do programa por meio da UART2. Ou seja, sempre que um botão fosse pressionado, transmitia-se um caracter específico pela UART2. Tal caracter era tratado no programa principal e em seu *“if”* eram setadas algumas variáveis que foram utilizadas como flags, além da execução de alguma instrução específica como, por exemplo, o acendimento dos leds quando a mensagem fosse transmitida.

Sempre que fosse necessário uma ação do computador 1 interferir no computador 2, o módulo de transmissão pela UART2 era utilizado. Tal sistema foi implementado na questão da autorização de impressão e na destruição da

mensagem. Note que nesses dois casos, os *push buttons* do computador 1 interferiam na mensagem armazenada no computador 2, e isso requeria que a transmissão da UART2 fosse utilizada.

3. **Implementação de um sistema que detecta que as placas não estão conectadas :**

Essa foi certamente a parte mais complexa do projeto, pois temos que mandar temporariamente um caractere pelo canal da UART2 e assegurar sua recepção. Caso este não seja recebido, devemos interromper a execução do código. Para a implementação de tal trecho do programa, usamos de uma variável contadora no próprio main que ao atingir CICLOS_TESTE (que foi definido com 100) ciclos envia um caractere especial (Usamos o ASCII 223). Após a recepção desse caractere, são setados alguns flags: o caractere_especial como 1; a verifica_conex como 1; e reiniciávamos o timer para o envio do próximo caractere.

Dessa forma, garantimos que sempre que fosse enviado um caractere especial, o programa não entrasse na rotina de colocar tal caractere na string da mensagem. Além de garantir o tratamento da rotina que verifica a queda de conexão, caso acontecesse.

● Possibilidades de expansão em Hardware e Software :

1. **Sistema de transmissão de dados Wireless[1] :**

Uma das limitações impostas pelo hardware utilizado no desenvolvimento deste projeto, foi o fato da conexão das placas ser feita via cabos. Dada a necessidade da onipresença da informação no mundo atual, vemos que uma abordagem mais interessante, a respeito da comunicação entre dois elementos eletrônicos, seria feita através de conexões que não dependessem das limitações físicas, ou pelo menos que tais limitações fossem definidas em um outro universo.

Uma transmissão wireless consiste numa conexão realizada sem a necessidade do uso de cabos (óticos, telefônicos, coaxiais, entre outros). Dessa forma, a intercomunicação entre os dois usuários da placa seria realizada a uma distância mais factível para a aplicação em mercado.

O uso da tecnologia WPAN, como o *Bluetooth*, poderia ser desenvolvida e empregada para distâncias pequenas, mas sem o uso de conectores físicos. Outra tecnologia, com larga utilização no mercado,é a Wifi. Wifi é uma rede sem fio, que utiliza de ondas de rádio para comunicação com internet e transmissão de dados entre dispositivos.

2. **Implementação de um servidor[2] :**

Outra técnica que elimina a necessidade do uso de cabos físicos é a criação de um servidor. Como um anexo ao projeto, a implementação de um servidor poderias ser realizada, apesar de tratar de linguagens não abordadas pela disciplina.

Um servidor web, por exemplo, permite que informações dispostas online (dados da digitação de um teclado, por exemplo) possam ser convertidas em palavras legíveis para o usuário. Além disso, a hospedagem na web permite que tais informações sejam armazenadas e acessadas quando necessário.

Tais implementações, como já mencionado, saem um pouco do escopo da disciplina, mas podem ser consideradas viáveis do ponto de vista de aprendizado. Isso permitiria que informações fossem passadas pela internet, eliminando a dependência dos cabos.

3. **Armazenagem das conversas localmente ou em nuvem[3] :**

Outra técnica muito interessante, ainda crescente no mercado, é a de armazenagem em nuvem. Tal tecnologia consiste em armazenar dados e informações em nuvem ou em discos virtuais disponibilizados por empresas específicas.

Esse tipo de conexão oferece segurança e um grande espaço para o usuário. Além de garantir que as informações sejam processadas com segurança, tal armazenamento permite que os dados, uma vez colocados, possam ser recuperados. No caso do projeto desenvolvido e explicado nesse relatório, os caracteres digitados pelos usuários poderiam ser colocados em uma nuvem e, a partir dela, serem acessados pelo outro usuário.

Tal implementação também eliminaria o uso de cabos. Entretanto, analogamente à armazenagem em um servidor web, o acesso à Internet seria de primordial importância, já que tais informações ficam armazenadas na rede.

● Conclusão :

O sistema implementado mostrou-se eficiente na realização das tarefas desejadas e seu desenvolvimento mostrou-se consistente com o escopo da disciplina. As possibilidades de expansão do projeto são imensas, e se realizadas de forma eficiente gerariam um sistema de mensagens confiável, robusto e funcional.

- **Fontes :**

[1]Autor desconhecido. Rede sem fio. Disponível em:<https://pt.wikipedia.org/wiki/Rede_sem_fio>. Acesso em 22/11/2015.
[2]JEANTY, J., tradução de TERRA, P. M. G. Como funciona um servidor? Disponível em: <http://www.ehow.com.br/funciona-servidor-como_6460/>. Acesso em 22/11/2015.
[3]Autor desconhecido. Entenda como funciona o armazenamento de dados na vuem. Disponível em : <http://www.lemonblue.com.br/blog/entenda-como-funciona-o-armazenamento-de-dados-na-nuvem/> . Acesso em 22/11/2015.