INSTITUTO FEDERAL DE GOIÁS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Alexandre Alves Trindade
Tiago de Morais Rosa Santos
Matheus Henrique Reis dos Santos

TRANSMISSÃO DE DADOS VIA RS485

Goiânia
2017

RESUMO

Este trabalho consiste na implementação de comunicação do tipo half duplex entre arduinos através do protocolo RS485. O protocolo do computador para o Arduino mestre é RS232. A comunicação é feita através de endereçamento, o mestre e escravo tem que ter o mesmo endereço para que possa ocorrer transmissão e recebimento de mensagens. Os requisitos de funcionamento deste trabalho incluem led indicador de recebimento de mensagens do escravo, led de recebimento de mensagens do mestre, botão que envia seu estado desligado ou ligado para o mestre e envio da leitura da temperatura do escravo para o mestre, utilizando o sensor LM35.

Palavras-chave: RS485. RS232. Arduino. Mestre. Escravo.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO COMPREENSIVA

O desafio deste trabalho é resolver a comunicação do fluxograma da figura 1, atendendo os requisitos detalhados a seguir.

Tabela 1. Funcionamento requisitado do projeto.

A forma de comunicação entre mestre e escravos pode ser via pooling ou via interrupção;
Os escravos devem ter um Led indicando a recepção de dados;
Os escravos devem ter um Led para indicar que receberam uma mensagem destinada a eles;
O mestre deve ter um led para indicar que recebeu uma informação;
Cada escravo deve ter um sensor de temperatura. Quando for feita a comunicação, o mestre deve receber essa informação. Essa informação (temperatura enviada) deve ser exibida na aplicação, indicando qual o escravo que enviou a informação. Através da aplicação, deve ser possível enviar uma mensagem para os escravos (de maneira individual), para acender um Led, acoplado a cada um dos módulos;
Cada escravo deve ter um botão. Quando for feita a leitura do escravo, a informação de boto ter sido pressionado ou não deve ser lida e indicada na aplicação.

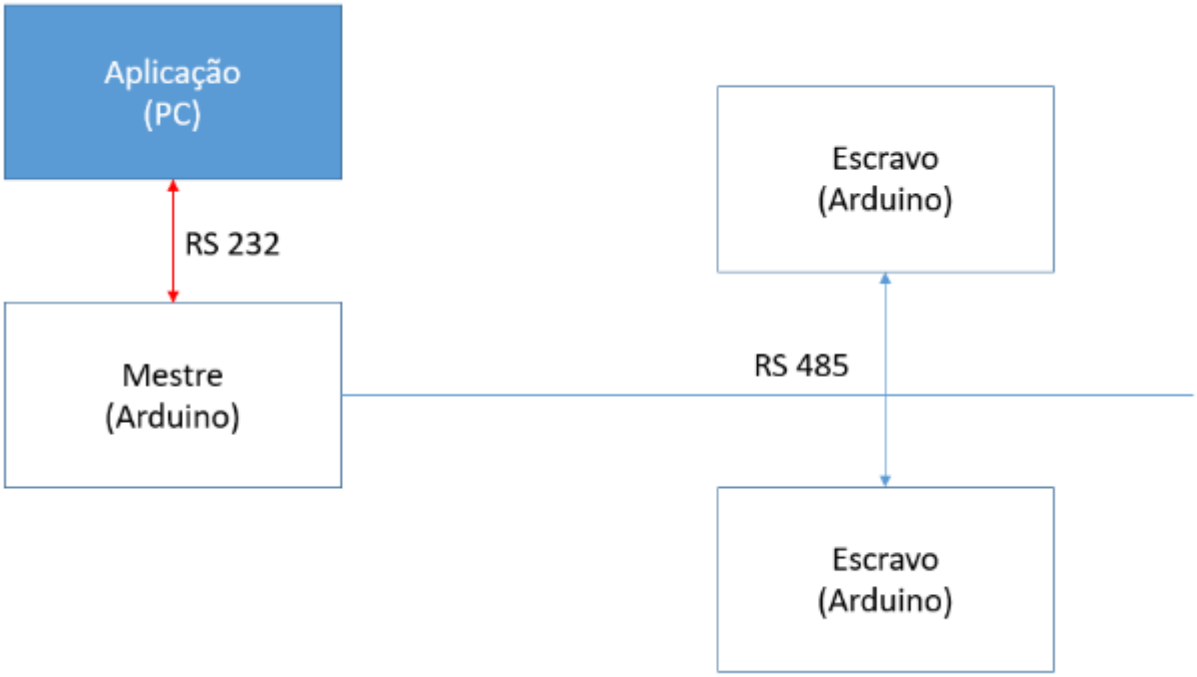

Figura 1. Esquema de configuração.

O padrão RS485 é um padrão de comunicação serial, também denominado EIA-485por ser desenvolvido ela EIA (Electronics Industry Association), que também desenvolveu os padrões de comunicação serial: RS-232 (EIA-232) e RS-422 (EIA-422). “RS” é a sigla para Recommended Standard (padrão recomendado).

O padrão RS-485 é baseado na transmissão diferencial de dados, através de um par de fios, que é ideal para transmissão em altas velocidades, longas distâncias e em ambientes propícios a interferência eletromagnética. Ele permite a comunicação entre vários elementos participantes em uma mesma rede de dados.

A transmissão diferencial de dados funciona da seguinte forma: qualquer transmissor RS-485 possui dois canais independentes conhecidos como A e B, que transmitem níveis de tensão iguais, porém com polaridades opostas (VOA e VOB ou simplesmente VA e VB). Por esta razão, é importante que a rede seja ligada com a polaridade correta. Embora os sinais sejam opostos, um não é o retorno do outro, isto, não existe um loop de corrente. Cada sinal tem seu retorno pela terra ou por um terceiro condutor de retorno, entretanto, o sinal deve ser lido pelo receptor de forma diferencial sem referência à terra ou ao condutor de retorno. Este sinal diferencial, lido em relação ao ponto central da carga, é que é interpretado como sinal de transmissão. Qualquer tensão maior que 200 mV é um nível alto ou “marca”. Uma tensão menor que -200 mV é um nível baixo ou espaço”. Níveis entre -200 mV e +200 mV são indefinidos e interpretados como ruído. [1]

O protocolo de comunicação serial RS-232 é um protocolo padrão usado na comunicação serial assíncrona. É o protocolo primário usado em linhas de modem. É o protocolo usado pelo MicroStamp11 quando ele se comunica com um PC host.

A Figura 2 mostra a relação entre os vários componentes em uma tinta serial. Esses componentes são o UART, o canal serial e a lógica da interface. Um chip de interface conhecido como o receptor/transmissor assíncrono universal ou UART é usado para implementar a transmissão de dados serial. O UART fica entre o computador host e o canal serial. O canal serial é a coleção de fios sobre os quais os bits são transmitidos. A saída do UART é um nível de lógica TTL/CMOS padrão de 0 ou 5 volts. A fim de melhorar a largura de banda, remover o ruído, e aumentar a gama, este nível lógico TTL é convertido em um nível de lógica RS-232 de -12 ou +12 volts antes de ser enviado no canal serial. Essa conversão é feita pela lógica de interface mostrada na Figura 2. Em seu sistema a lógica de interface é implementada pelo selo de comunicação.

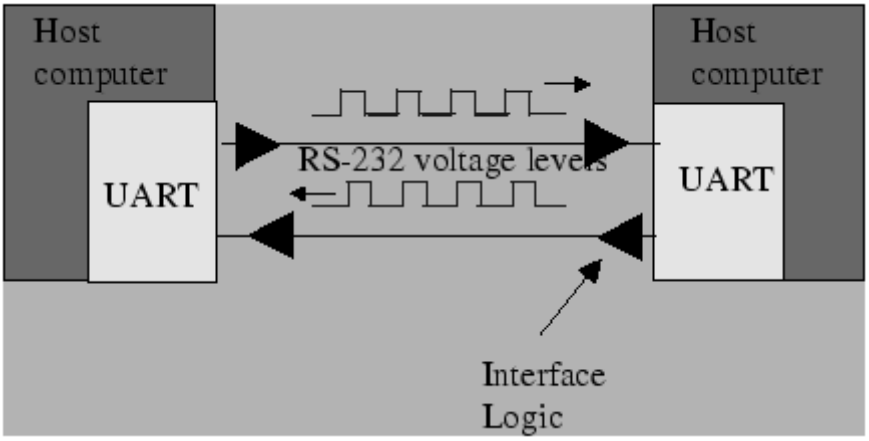

Figura 2. Link assíncrono (RS232).

Fonte: http://controls.ame.nd.edu/microcontroller/main/node24.html

Um frame é um pacote completo indivisível de bits. Um quadro inclui ambas as informações (por exemplo, dados e caracteres) e sobrecarga (por exemplo, bit de início, verificação de erros e bits de parada). Em protocolos seriais assíncronos, como RS-232, o frame consiste em um bit de início, sete ou oito bits de dados, bits de paridade e bits de parada. Um diagrama de temporização para um quadro RS-232 consistindo em um bit de início, 7 bits de dados, um bit de paridade e dois bits de parada é mostrado abaixo na Figura 3. Observe que a estrutura exata do quadro deve ser acordada pelo transmissor e pelo receptor antes que o link comum deve ser aberto.[2]

Figura 3. Frame RS232 (1 start bit, 7 data bits, 1 parity bits, and 2 stop bits.)

Fonte: http://controls.ame.nd.edu/microcontroller/main/node24.html

1.1 JUSTIFICATIVA

Este trabalho se enquadra nos tópicos divididos de redes industriais, onde estudamos os diferentes tipos de protocolos e formas de comunicação. Entre a opção de utilizar o protocolo ZigBee, optamos em implementar a comunicação entre arduinos mestre/escravo devido ao custo ser consideravelmente menor do módulo MAX485, que usamos neste trabalho.

1.2.1 Objetivo geral

Estabelecer a comunicação entre dois arduinos utilizando o protocolo RS485.

1.2.2 Objetivos específicos

Acender o led de recebimento de mensagem do escravo, e também o led quando o mestre recebe mensagens. Um botão de envio de mensagem para o mestre, e implementar a leitura da temperatura.

2 COMPARAÇÃO PROTOCOS RS235 E RS485

A figura 4 representa em forma de tabela comparativa as diferenças da comunicação serial RS232, RS422 e RS485.

Especificações	RS-232	RS-422	RS-485
Modo de Operação	Desbalanceado	Diferencial	Diferencial
Número Total de Transmissores e Receptores em Uma Linha (Um transmissor ativo por um tempo para redes RS-485)	1 Transmissor 1 Receptor	1 Transmissor 10 Receptores	32 Transmissores 32 receptores
Comprimento máximo do Cabo	50 ft (2500 pF)	4000 ft	4000 ft
Máxima Taxa de Dados (40 ft - 4000 ft para RS-422/RS-485)	160 kbits/s (pode ser até 1MBit/s	10 MBit/s	10 MBit/s

Figura 4. Comparação da comunicação serial.

Fonte: http://digital.ni.com/public.nsf/allkb/DE153F74C4BF3AD8862576AB006C1AAF

MATERIAIS UTILIZADOS

- **3 LEDs**
- **4 Resistores**
- **2 Arduinos UNO**
- **2 Protoboards**
- **1 Botoeira**
- **Fios**
- **2 Cabos USB AB**
- **Módulo Conversor TTL Para RS-485 Arduino MAX485**

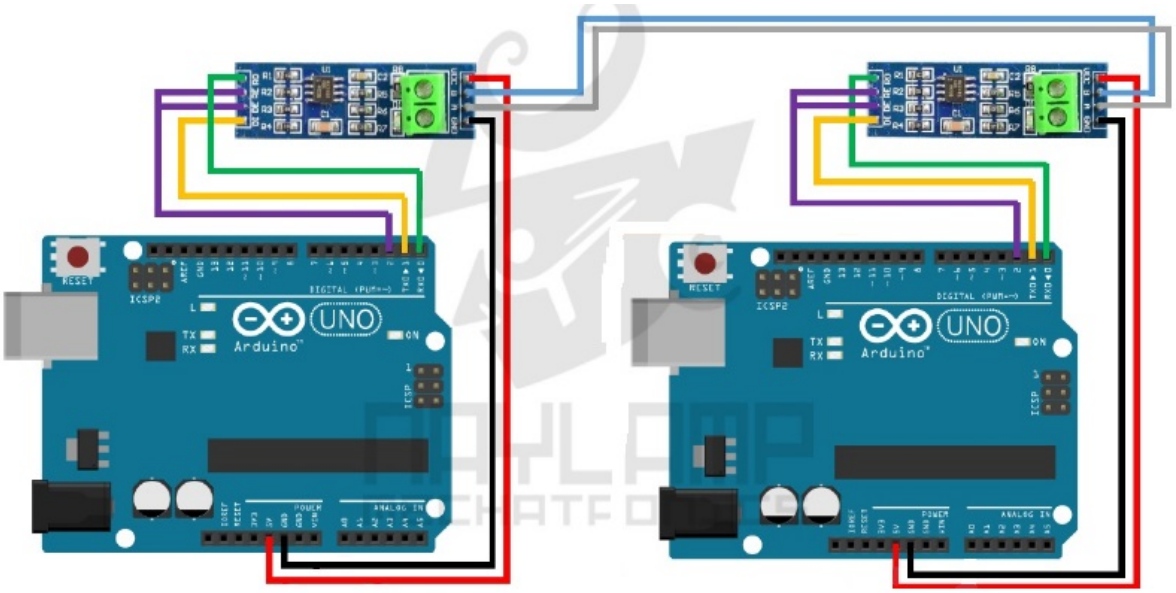

Figura 5. Esquema de montagem.

Fonte: http://www.naylampmechatronics.com/blog/37_Comunicaci%C3%B3n-RS485-con-Arduino.html

MONTAGEM

As conexões entre os Arduinos e os Módulos RS-485 foram idênticas tanto para o mestre quanto para o escravo. O GND do módulo RS-485 foi ligado ao GND do Arduino, o VCC do módulo RS-485 foi ligado à saída de 5V do Arduino. O pino DE e RE foram curto circuitados e ligados à porta 2 do Arduino. O pino DI (Data input) foi ligada à saída 1 (TX0) do Arduino. A saída 0 (RX0) foi conectada ao pino RO (Receiver Output).

Após ligarmos cada Arduino a um Módulo RS-485 da forma citada acima, ligamos o pino A de um módulo RS-485 ao pino A do outro módulo RS-485 e posteriormente foi feito o

mesmo ao pino B, ou seja, conectou-se o pino B de um módulo ao pino B do outro módulo RS-485.

Ao Arduino do escravo foi conectado um LED à saída 8, um LED à saída 6 e um botão à saída 9. No Arduino do mestre foi conectado um LED à saída 8 e um LED à saída 7.[3]

DESCRIÇÃO DO CÓDIGO

Devido à limitação do número de módulos RS485 disponíveis (o grupo só conseguiu adquirir dois módulos), o sistema contou com apenas um mestre e um escravo. No entanto, o código foi desenvolvido realizando o endereçamento do escravo, de modo que ele só recebe a mensagem do mestre caso o endereço enviado na mensagem seja compatível com o seu endereço. Assim, nosso software dá suporte para o desenvolvimento de um sistema que permita o acréscimo de novos escravos. Os códigos desenvolvidos estão em anexo no final do relatório.

Estrutura da mensagem

Tabela 2. Vetor de caracteres.

Caractere de início	Endereço do escravo	Caractere de Função	Valor da mensagem	Caractere de Fim

Funções:

A - ligar led
L - ler sensor de temperatura
B - ler estado do botão

CÓDIGO DO ESCRAVO

Primeiramente, declara-se as variáveis envolvidas no projeto, como o número das portas e o próprio endereço do escravo. Na função setup, indicamos se as portas trabalham em INPUT ou OUTPUT e definimos alguns estados de inicialização.

Dentro do loop, o primeiro passo é verificar se há transmissão de mensagens através do comando *Serial.available()*. Caso a mensagem se inicie com o caractere I, lemos o restante da string (este caractere indica que a mensagem se iniciará). Passado este primeiro passo, verifica-se o código de escravo enviado pelo mestre. O restante da mensagem só é processada caso haja

correspondência entre endereço próprio e endereço enviado pelo mestre. Feita a verificação, ligamos o led de recebimento de mensagem. Em seguida, lemos o caractere de função e guardamos na variável *funcion*.

A próxima etapa do algoritmo consiste em um conjunto de *ifs*. Caso a função solicitada seja “A”, ligamos o led do pino *ledPin* se a mensagem enviada após o caractere “A” for igual a 1. Se não for igual a 1, o led é apagado. O último caractere recebido deve ser um “F”, que indica fim de mensagem.

Caso a função solicitada seja “B” lemos o estado do pino associado ao botão: a porta *botao*. Feita a leitura, habilitamos o escravo para se tornar um transmissor, alterando a porta 2 para HIGH. Em seguida, enviamos o pacote de caracteres contendo os caracteres “i”, “mydireccion” (endereço do escravo), “,”, o caractere que indica o estado do pino do botão e o caractere “f”, indicando fim de mensagem. Por fim, voltamos a definir o escravo como receptor alterando o status do pino 2.

Caso a função solicitada seja “L”, fazemos o mesmo processo implementado para a função “B”, com a diferença de mandar um valor de 0 a 1023 associado à temperatura lida pelo LM35 ao invés de enviar o status do botão.

CÓDIGO DE MESTRE

Nas linhas iniciais declaramos as variáveis utilizadas e na função setup definimos estados de inicialização, assim como no escravo.

O mestre faz o controle de tempo e é o responsável por fazer as solicitações, portanto, ele passa a maior parte do tempo como emissor de mensagens. Ele só opera quando receptor quando solicita alguma resposta dos escravos. Da mesma forma, os escravos operam principalmente como receptores, e só enviam a mensagem quando necessário. Após enviar mensagens, eles imediatamente alteram o estado do pino 2 para LOW, atuando como receptor novamente.

No loop, primeiramente, o mestre solicita o ligamento de um led no escravo enviando o valor *envio* = 0. Neste caso, ele não espera respostas do escravo.

Posteriormente, ele solicita a leitura de temperatura para o escravo, enviando no campo de função o caractere “L”. A partir de então ele opera como receptor e espera a resposta do escravo. Chegada a resposta, ele identifica o caractere “i” de início, liga o led de recebimento, verifica o endereço do escravo e armazena a variável de dados de temperatura recebida. Ao

chamar a função *funcion*, converte-se o valor de 0 a 1023 para o valor efetivo em °C. No próximo passo, é mostrado na tela a temperatura do escravo.

No próximo loop, o mestre envia para o escravo a opção de função “B”. O mesmo procedimento feito para a leitura de temperatura é feito para o estado atual do botão, e de acordo com esse estado é retornado ao usuário “ligado” ou “desligado”.

Todas as mensagens mostradas contêm o endereço do escravo que enviou a informação.

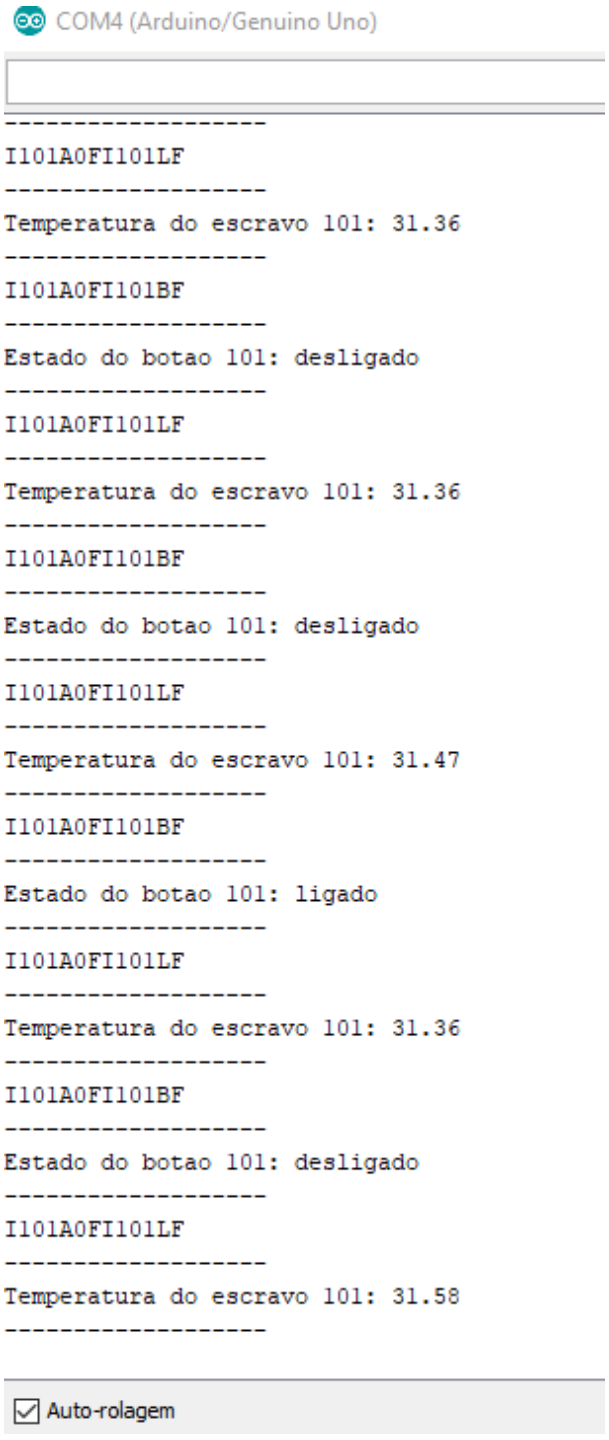

Figura 6. Mensagens lidas no Serial Monitor do Mestre.

3 CONCLUSÕES E PERSPECTIVAS

Com base nos resultados alcançados, descritos neste trabalho, a comunicação mestre escravo foi realizada com sucesso. Os leds indicativos de recebimento e transmissão de mensagens, guiam o usuário para entender o que está acontecendo nesse tipo de comunicação. A temperatura é lida continuamente pelo mestre, e o escravo tem a capacidade de enviar mensagens, como o botão do nosso projeto, para o mestre.

O algoritmo é feito na comunicação tipo pooling, e para que aconteça com sucesso o envio e recebimento de mensagens o escravo e mestre tem que possuir o mesmo número de endereço. Para trabalhos futuros recomendemos a implementação de mais de um escravo para testar essa capacidade que foi prevista no algoritmo desse projeto, porém testada até essa data com um escravo.

REFERÊNCIAS

[1] http://olaria.ucpel.tche.br/autubi/lib/exe/fetch.php?media=padrao_rs485.pdf
[2] http://controls.ame.nd.edu/microcontroller/main/node24.html
[3] http://www.naylampmechatronics.com/blog/37_Comunicaci%C3%B3n-RS485-con-Arduino.html

ANEXO A – CÓDIGO DO ESCRAVO

```
const int ledPin =  8; // Numero do Pino para o LED

const int EnTxPin =  2;  // HIGH:TX y LOW:RX

const int mydireccion =101; //codigo do escravo

const int botao = 9;

const int ledDestino = 6;


boolean liga;

int reading;

int tempPin = A0;

int estado;


void setup() {

  analogReference(INTERNAL);

  Serial.begin(9600);

  Serial.setTimeout(100);  //estabelecemos um tempo de espera de 100ms

  pinMode(EnTxPin, OUTPUT);

  pinMode(ledPin, OUTPUT);

  pinMode(botao, INPUT);

  pinMode(ledDestino, OUTPUT);

  digitalWrite(ledPin, LOW);

  digitalWrite(EnTxPin, LOW); //RS485 como receptor

}


void loop() {
```

```
void loop() {
 if(Serial.available())
 {
  if(Serial.read()=='I') //Se recebermos o inicio da mensagem
  {
   int direccion=Serial.parseInt(); //recibemos la direção
   if(direccion==mydireccion) //verificação do endereço
    {
     digitalWrite(ledDestino, HIGH);
     char funcion=Serial.read(); //ler o caractere de função

     if(funcion=='A'){
      int envio=Serial.parseInt(); // receber comando do led
      if(Serial.read()=='F') //Se o fim de mensagem é o correto
          {
           if(envio==1) //verificamos o valor
            {
             digitalWrite(ledPin, HIGH);

            }
            else{digitalWrite(ledPin, LOW);}
          }
     }
```

```
if(funcion=='B'){
  if(Serial.read()=='F') //Se o fim da mensgem é o correto
      {
       if(digitalRead(botao) == HIGH){
       estado = 1;
       } else {estado = 0;}


        digitalWrite(EnTxPin, HIGH); //rs485 como transmissor
        Serial.print("i"); //inicio da mensagem
        Serial.print(mydireccion); //direção
        Serial.print(","); //temperatura
        Serial.print(estado);
        Serial.print("f"); //fimda mensagem
        Serial.flush(); //Esperamos até que se envie os dados
        digitalWrite(EnTxPin, LOW); //RS485 como receptor
      }
}
```

```
        //leitura de temperatura
        if(funcion=='L')
          {
            if(Serial.read()=='F') //Se o fim da mensagem é o correto
             {


               reading = analogRead(tempPin);



               digitalWrite(EnTxPin, HIGH); //rs485 como transmissor
               Serial.print("i"); //inicio da mensagem
               Serial.print(mydireccion); //direção

               Serial.print(","); //temperatura
               Serial.print(reading);
               Serial.print("f"); //fim da mensagem
               Serial.flush(); //Esperamos até que se enviem ps dados
               digitalWrite(EnTxPin, LOW); //RS485 como receptor
             }
          }
          delay(100);
          digitalWrite(ledDestino, LOW);
       }
    }


  }


   delay(10);
}
```

ANEXO B – CÓDIGO DO MESTRE

```
const int ledPin =  8; // Numero del pin para el Led

const int EnTxPin =  2; // HIGH:TX y LOW:RX

const int ledRecebe = 7;


String endereco = "101";

int enderecoi = 101;

float tempC;

char opcao;

String bot;

int esclavo;

int dato;

int estado;


void setup() {

  opcao = 'l';

  Serial.begin(9600);

  Serial.setTimeout(100);

  pinMode(ledPin, OUTPUT);

  pinMode(EnTxPin, OUTPUT);

  pinMode(ledRecebe, OUTPUT);

  digitalWrite(ledPin, LOW);

  digitalWrite(EnTxPin, HIGH); //RS485 como Transmissor

}
```

```
void loop() {
 int envio = 0;
 Serial.print("I"); //inicio da mensagem
 Serial.print(endereco);//direçao do escravo
 Serial.print("A"); //função A para indicar que vamos ascender o led do escravo
 Serial.print(envio); //angulo  o dato
 Serial.print("F"); //fim da mensagem

 //---solicitamos uma leitura do escravo----------
 if(opcao == 'l'){
 Serial.print("I"); //inicio da mensagem
 Serial.print(endereco);//direção do escravo
 Serial.print("L"); //L para indicar que vamos leer o sensor
 Serial.print("F"); //fim da mensagem
 Serial.flush();    //Esperamos até que se enviem os dados

 if(opcao == 'b'){
 // solicitar estado do botao
 Serial.print("I"); //inicio da mensagem
 Serial.print(endereco);//direção do escravo
 Serial.print("B"); //L para indicar que vamos a Leer o sensor
 Serial.print("F"); //fimda mensagem
 Serial.flush();    //Esperamos até que se envie os dados

 // resposta do escravo
 digitalWrite(EnTxPin, LOW); //RS485 como receptor
```

```
if(Serial.find("i")) //esperamos o incicio da mensagem
{
   esclavo=Serial.parseInt();  //recebemos a direção do escravo
   estado=Serial.parseInt(); //receber o dado (dato)

   digitalWrite(ledRecebe, HIGH);
   delay(100);
   digitalWrite(ledRecebe, LOW);

   //float temperatura=Serial.parseInt();  //recebe temperatura do escravo
   if(Serial.read()=='f'&&esclavo==enderecoi) //Se fim de mensagem e direção são os corretos
   {
     //temp(temperatura, esclavo); //função temp
     bot = verificarBotao(estado);   //realizamos a ação com o dado(dato) recebido
   }

}
digitalWrite(EnTxPin, HIGH); //RS485 como Transmissor

}
```

```
 if(opcao == 'l')
 {
    Serial.println(" ");
 Serial.println("--------------------");
 Serial.print("Temperatura do escravo ");
 Serial.print(esclavo);
 Serial.print(": ");
 Serial.println(tempC);
 Serial.println("--------------------");
  opcao = 'b';
 }
 }
 else{
    Serial.println(" ");
    Serial.println("--------------------");
 Serial.print("Estado do botao ");
 Serial.print(esclavo);
 Serial.print(": ");
 Serial.println(bot);
 Serial.println("--------------------");
  opcao = 'l';
 }

 delay(1000);
}
```

```
float funcion(int dato)
{
  float tempC = dato / 9.31;
  return tempC;
}

String verificarBotao(int estado)
{
  String bot = "abcd";
  if(estado == 1){
    bot = "ligado";
  }
  if(estado == 0)
  {bot = "desligado";}
  return bot;
}
```