Universidade Federal de Uberlândia
Faculdade de Engenharia Mecânica
Redes Industriais
Professor José Jean-Paul Zanlucchi de Souza Tavares

RELATÓRIO EXERCÍCIO PRÁTICO – GRUPO 01
COMUNICAÇÃO INTEGRADA ENTRE DISPOSITIVOS

GUILHERME DE ALMEIDA GONÇALVES TIAGO - 11821EMT022
ALISSON CARVALHO VASCONCELOS - 11511EMT016

Uberlândia – abril de 2023

Sumário

1. Introdução

Este grupo ficou com a sequência 1 de atuação distinta, conforme se segue:

Grupo 01: A+{A-//B+}A+{B-//A-}

O esquema eletropneumático completo, isto é, composto de 1 Botão, 2 cilindros de dupla ação (A e B), dois fins de curso (A0/A1 B0/B1), duas válvulas simples solenoide 5x2 com retorno por mola e 5 Arduinos (numerados de 1 a 5), conforme mostra a Figura 1.

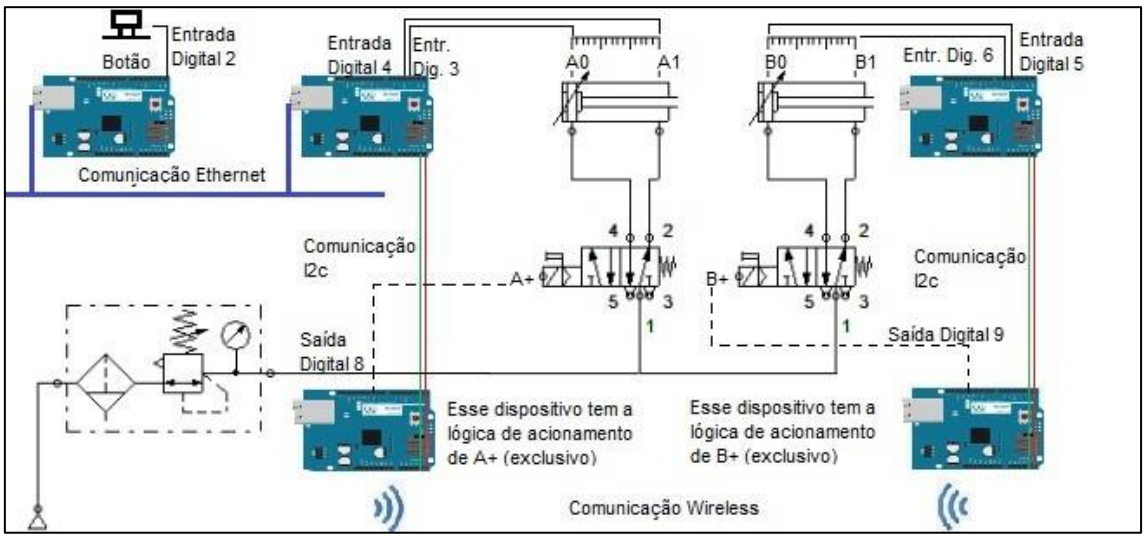

Figura 1: Esquema eletropneumático completo.

2. Desenvolvimento

2.1 Configuração dos cabos e arduinos

O botão deve estar conectado na entrada digital 2 do Arduino 1, que se comunica com o Arduino 2 por meio de rede Ethernet. Os fins de curso, A0 e A1, devem estar, respectivamente, nas entradas digitais 3 e 4 do Arduino 2, que será o escravo numa rede I2C com um mestre, que será o Arduino 3, responsável pela lógica de acionamento da válvula A+ (saída digital 8). Por outro lado, os outros fins de curso, B0 e B1, deverão estar, respectivamente, nas estradas digitais 5 e 6 do Arduino 4, que será escravo numa rede I2C do Arduino 5, que está vinculado à lógica de atuação da válvula B+ (que é a saída digital 9). Os arduinos que estão relacionados com as válvulas devem se comunicar via Xbee (sem fio, wireless).

2.2 Sistema I2C (mestre-escravo)

Comunicação I2C: no laboratório foi realizada a configuração do arduinos (Mestre/Escravo) para acionamento do pistão via comunicação I2C onde foram feito duas configurações:

- Mestre Recebedor/Escravo Emissor: o mestre solicita as informações do escravo (1S0, 1S1 e do botão) e após o acionamento do botão o mestre aciona (Liga ou Desliga) o rele (aciona a válvula simples 5/2 com solenoide e retorno por mola).

- Mestre Emissor/Escravo Recebedor: o mestre realiza a leitura dos sensores (1S0, 1S1) e do botão, e após o acionamento do botão o mestre requisita o acionamento (Liga ou Desliga) do relé (aciona a válvula simples 5/2 com solenoide e retorno por mola) do escravo.

Foram feitos 4 códigos para essa parte: Escravo_recive.ino, Escravo_send.ino, Mestre_recive.ino, Mestre_send.ino, apresentados nas imagens a seguir.

Escravo_recive.ino:

Escravo_recive

```
#include <Wire.h> //Inclusão da biblioteca Wire.h
#define slave 2 //Definição uma constante do endereço do escravo
#define rele 7  //Definição uma constante do rele
#define avanco 0  //Definição uma constante de avanco
#define recuo 1  //Definição uma constante de recuo

void setup() {
  Serial.begin(9600);             //Inicio da comunicacao Serial e define a velocidade de comunicação
  Wire.begin(slave);              //Inicio da comunicação via bus entre mestre/escravo
  Wire.onReceive(receiveEvent);   //Definição de quando solicitado chamamos a função reveiveEvent

  pinMode(rele, OUTPUT);          //Definição da pinagem do rele como saida
  digitalWrite(rele,recuo);  //Inicio solenoide ativado (Recuado)
}

void loop() { }

//Função de recebimento de dados do mestre
void receiveEvent(int howMany) {
  Serial.println("Conexão estabelecida.")
  //Loop de verificação de dados na bus para recebimento
  while(Wire.available()) {
    char buf = Wire.read();            //Atribuição do valor do rele buffer enviado pelo mestre
    Serial.print("Dado recebido: {")
    Serial.print(buf, DEC);
    Serial.println("}")

    digitalWrite(rele, buf);      //Escrita no pino do rele de 0V/5V para acionamento do solenoide (avanco/recuo)
  }
  Serial.println("Fim de Transmissão.")
}
```

Escravo_send.ino:

Escravo_send

```
#include <Wire.h>                 //Inclusão da biblioteca Wire.h
#define slave 2                   //Definição uma constante do endereço do escravo
#define bt 2                      //Definição uma constante do botão
#define a0 4                      //Definição uma constante do sensor A0
#define a1 6                      //Definição uma constante do sensor A1

char buf[3] = {0,0,0};            //Definição de um vetor de 3 caracteres

void setup() {
  Serial.begin(9600);
  Wire.begin(slave);              //Inicio da comunicação via bus entre mestre/escravo
  Wire.onRequest(requestEvent);   //Definição de quando solicitado chamamos a função requestEvent

  pinMode(bt, INPUT_PULLUP);      //Definição da pinagem do botão como entrada
  pinMode(a0, INPUT_PULLUP);      //Definição da pinagem do sensor A0 como entrada
  pinMode(a1, INPUT_PULLUP);      //Definição da pinagem do sensor A1 como entrada
}

void loop() {
  buf[0] = digitalRead(bt);       //Atribuição ao vetor 0 do buffer a leitura do botão
  buf[1] = digitalRead(a0);       //Atribuição ao vetor 1 do buffer a leitura do sensor A0
  buf[2] = digitalRead(a1);       //Atribuição ao vetor 2 do buffer a leitura do sensor A1
}

//Função de envio de dados para o mestre
void requestEvent(){
  /*
  write(buf,len) - escrita de byte
  buf - vetor dados para envio no formato de bytes
  len - tamanho do buffer a ser enviado
  */
  Wire.write (buf, 3);            //Escrita de dados em formato de bytes com 3 buffer a ser enviado para o mestre
}
```

Mestre_recive.ino:

Mestre_recive

```
#include <Wire.h>                 //Inclusão da biblioteca Wire.h
#define slave 2                  //Definição uma constante do endereço do escravo
#define rele 7                    //Definição uma constante do rele

char bt, a0, a1;          //Definição de variáveis como caractere
volatile bool flag = false;       //Definição de flag para execução

void setup() {
  Wire.begin();                   //Inicio da comunicação via bus entre mestre/escravo
  Serial.begin(9600);             //Inicio da comunicacao Serial e define a velocidade de comunicação

  pinMode(rele, OUTPUT);          //Definição da pinagem do rele como saida
  digitalWrite(rele,HIGH);
}

void loop() {
  Wire.requestFrom(slave,3);      //Requisita 3 bytes do escravo

  //Loop de verificação de dados para recepção de dados
  while(Wire.available()) {
    bt = Wire.read();             //Atribuição do valor do botão na leitura dos dados enviados pelo escravo
    a0 = Wire.read();             //Atribuição do valor do sensor A0 na leitura dos dados enviados pelo escravo
    a1 = Wire.read();             //Atribuição do valor do sensor A1 na leitura dos dados enviados pelo escravo

    //Imprime no monitor serial os valores dos sensores A0 e A1
    Serial.print("Botão: ");
    Serial.print(bt, DEC);
    Serial.print("\tSensor A0: ");
    Serial.print(a0, DEC);
    Serial.print("\tSensor A1: ");
    Serial.println(a1, DEC);
```

```
32
33    if (!flag && bt && a0 == 1) {
34      flag = true;
35    }
36  }
37
38  //Verificação se a variável aux possui valor
39  if (flag) {
40    //Condição para acionamento de avanço ou recuo do pistão quando o sensor for acionado.
41    if (a0 == 1 && a1 == 0) {
42      digitalWrite(rele, LOW);    //Escrita no pino do rele de 0V para acionamento do solenoide (avanco)
43    }
44    else if(a0 == 0 && a1 == 1) {
45      digitalWrite(rele, HIGH);   //Escrita no pino do rele de 5V para acionamento do solenoide (recuo)
46      flag = false;
47    }
48  }
49
50  delay(500);                     //Atraso de 0,5 segundos
51}
```

Mestre_send.ino:

Mestre_send

```
1  #include <Wire.h> //Inclusão da biblioteca Wire.h
2  #define slave 2 //Definição uma constante do endereço do escravo
3  #define pinbt 2  //Definição uma constante do botão
4  #define pina0 4  //Definição uma constante do sensor A0
5  #define pina1 6  //Definição uma constante do sensor A1
6
7  int aux = 0;  //Definição de uma variável inteira
8  char bt, a0, a1;  //Definição das variáveis de leitura
9  volatile bool flag[2] = {false,false};  //Definição de flags
10
11 void setup() {
12   Serial.begin(9600); //Inicio da comunicacao Serial e define a velocidade de comunicação
13   Wire.begin(); //Inicio da comunicação via bus entre mestre/escravo
14
15   pinMode(pinbt, INPUT_PULLUP);  //Definição da pinagem do botão como entrada
16   pinMode(pina0, INPUT_PULLUP);  //Definição da pinagem do sensor A0 como entrada
17   pinMode(pina1, INPUT_PULLUP);  //Definição da pinagem do sensor A1 como entrada
18 }
19
20 void loop() {
21   Wire.beginTransmission(slave);  //Inicia a Transmição com o escravo
22   Serial.print("Conexão Estabelecida."); //Imprime a conectividade
23   bt = digitalRead(pinbt); //Atribuição do valor de leitura do botão
24   a0 = digitalRead(pina0); //Atribuição do valor de leitura do sensor A0
25   a1 = digitalRead(pina1); //Atribuição do valor de leitura do sensor A1
26
27   /*
28   //Imprime no monitor serial os valores dos sensores A0 e A1
29   Serial.print("Botão: ");
30   Serial.print(bt,DEC);
31   Serial.print("Sensor A0: ");
32   Serial.print(a0,DEC);
33   Serial.print("\tSensor A1: ");
34   Serial.print(a1,DEC);
```

```
  */

  //Verificação quantas vezes o botão foi acionado
  if (bt && !flag[0]){
    flag[0] = true; //Atribuição da indicação de que o botão foi precionado
  }
  if (bt == 0 && flag[0]) {
    aux++; //Incrementa da váriavel uma unidade
    flag[0] = false;  //Atribuição da indicação de que o botão foi liberado
  }

  /*
  //Imprime o valor do numero de vezes precionada
  Serial.print("Contador: ");
  Serial.print(aux, DEC);
  */

  //Comando para acionamento do solenoide
  if (aux>0){
    //Condição para acionamento de avanço ou recuo do pistão quando o sensor for acionado.
    if (a0 == 1 && a1 == 0 && !flag[1]){
      /*
      write(buf,len) - escrita de byte
      buf - vetor dados para envio no formato de bytes
      len - tamanho do buffer a ser enviado
      */

      Wire.write(0);  //Escrita de dados em formato de bytes a ser enviado para o escravo
      Serial.println("Dado Enviado: Execução Avanço"); //Imprime informação de que o dado foi enviado
      flag[1] = true; //Atribuição de que o acionamento foi realizado (Avanço)
    }
    else if(a1 == 0 && a0 == 1 && flag[1]){
      /*
      write(buf,len) - escrita de byte
      buf - vetor dados para envio no formato de bytes
      len - tamanho do buffer a ser enviado
      */
      Wire.write(1);  //Escrita de dados em formato de bytes com 1 buffer a ser enviado para o escravo
      aux--;  //Decrementa da váriavel uma unidade
      Serial.println("Dado Enviado: Execução Recuo"); //Imprime informação de que o dado foi enviado
      flag[1] = false; //Atribuição de que o acionamento foi realizado (Recuo)
    }
  }

  Serial.println("Fim Transmissão."); //Imprime que a transmissão foi finalizada
  Wire.endTransmission(); //Finaliza a transmissão
  delay(100);
}
```

2.3 Sistema Wireless (Xbee)

Comunicação Xbee: no laboratório foi realizada a configuração do arduinos (Emissor/Receptor) para acionamento do pistão via comunicação Ethernet onde o emissor envia as informações dos sensores (1S0, 1S1) e do botão, e após o acionamento do botão o receptor aciona (Liga ou Desliga) o relé (aciona a válvula simples 5/2 com solenoide e retorno por mola).

Para essa parte foram montados dois códigos: Receptor.ino e Emissor.ino, apresentados a seguir.

Emissor.ino:

Emissor

```
#include <XBee.h>
#define pinbt 2  //Definição de uma constante do botão de acionamento
#define pina0 5  //Definição de uma constante do sensor A0
#define pinal 3  //Definição de uma constante do sensor A1
#define avanco 1 //Definição de uma constante de avanco
#define recuo 0 //Definição de uma constante de avanco

volatile bool flag = false;
unsigned long tempo_disparo = 0;
int ultestbt;

XBee xbee = XBee();
uint8_t payload[]={0};
XBeeAddress64 add64 = XBeeAddress64(0x0013A200,0x40C1B1F0);
ZBTxRequest data = ZBTxRequest(add64,payload,sizeof(payload));
ZBTxStatusResponse txStatus = ZBTxStatusResponse();

volatile bool flag1 = false, flag2 = false, flag3 = false; //Definição de flags
char bt, a0, a1, texto; //Definição das variáveis como caractere
volatile unsigned long tempo = 0;

void setup(){
  Serial.begin(9600);
  xbee.setSerial(Serial);

  pinMode(pinbt, INPUT_PULLUP);
  pinMode(pina0, INPUT_PULLUP);
  pinMode(pinal, INPUT_PULLUP);
}

void loop(){
  sensor();

  //Verifica e bloqueia o acionamento do botão
  if (bt == 1 && flag1 && !flag2){
    flag1 = false;
  }
  if (bt == 0 && !flag1 && !flag2){
    flag2 = true;
  }

```

```
  if (flag2){
    //Condição para acionamento de avanço ou recuo do pistão quando o sensor for acionado.
    if (a0 == 1 && a1 == 0){
      payload[0] = avanco;
      xbee.send(data);
      flag3 = true;
    }
    while (a1 != 1&&flag3) {
      sensor();
    }
    if (flag3) {
      tempo = millis();
    }
    flag3 = false;
    if(a0 == 0 && a1 == 1 && (millis() - tempo) >= 5000) {
      payload[0] = recuo;
      xbee.send(data);
      flag1 = true;
      flag2 = false;
    }
  }
}

void sensor() {
  bt = digitalRead(pinbt); //Atribuição do valor da leitura do botão
  a0 = digitalRead(pina0); //Atribuição do valor da leitura do sensor A0
  a1 = digitalRead(pina1); //Atribuição do valor da leitura do sensor A1

  Serial.print("Botão: ");
  Serial.print(bt, DEC);
  Serial.print("\tSensor A0: ");
  Serial.print(a0, DEC);
  Serial.print("\tSensor A1: ");
  Serial.print(a1, DEC);
}
```

Receptor.ino:

Receptor

```
#include <XBee.h> // inclui a biblioteca
#define pinrele 7

XBee xbee = XBee(); // Cria um objeto XBee

//Cria um objeto ZBRx na variavel rx
ZBRxResponse recive =ZBRxResponse();

void setup(){
  Serial.begin(9600);
  xbee.setSerial(Serial);
  pinMode (pinrele, OUTPUT);
}

void loop(){
  uint8_t resposta ;
  xbee.readPacket();

  //Checando se o pacote foi recebido
  if(xbee.getResponse().isAvailable()){
    if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE){
      xbee.getResponse().getZBRxResponse(recive);
      resposta = recive.getData(0);
    }
  }

  digitalWrite(pinrele, resposta);
}
```

2.4 Sistema Ethernet (com fio)

Comunicação Ethernet: no laboratório foi realizado a configuração do arduinos (Emissor/Receptor) para acionamento do pistão via comunicação Ethernet onde o emissor envia as informações dos sensores (1S0, 1S1) e do botão, e após o acionamento do botão o receptor aciona (Liga ou Desliga) o relé (aciona a válvula simples 5/2 com solenoide e retorno por mola).

Foram montados dois códigos: Receptor.ino e Emissor.ino, apresentados a seguir.

Emissor.ino:

Emissor

```
#include <SPI.h>  //Inclusão da biblioteca SPI.h
#include <Ethernet.h> //Inclusão da biblioteca Ethernet.h
#include <Udp.h>  //Inclusão da biblioteca Udp.h
#define port 8888 //Definição da porta de comunicação
#define pinbt 2  //Definição de uma constante do botão de acionamento
#define pina0 5  //Definição de uma constante do sensor A0
#define pinal 3  //Definição de uma constante do sensor A1
#define avanco 1 //Definição de uma constante de avanco
#define recuo 0 //Definição de uma constante de avanco

EthernetUDP Udp;
byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x64, 0x50};  //Atribuição do MAC Address do arduino emissor
byte ip[] = {192,168,1,131};  //Atribuição do IP Address do arduino emissor
byte remoteip[] = {192,168,1,151};  //Atribuição do IP Address do arduino receptor

char packetBuffer[UDP_TX_PACKET_MAX_SIZE];  //Definição do tamanho do packetBuffer (24 byte)
volatile bool flag1 = false, flag2 = false, flag3 = false; //Definição de flags
char bt, a0, a1, texto; //Definição das variáveis como caractere
int count = 0;  //Definição de uma variável incremental
volatile unsigned long tempo = 0;

void setup() {
  Serial.begin(9600); //Inicialização da porta Serial
  iniEthernet();  //Inicialização do adaptador de rede

  pinMode(pinbt,INPUT_PULLUP); //Definição da pinagem do botão como entrada
  pinMode(pina0,INPUT_PULLUP); //Definição da pinagem do sensor A0 como entrada
  pinMode(pinal,INPUT_PULLUP); //Definição da pinagem do sensor A1 como entrada
}

void loop() {
  sensor();

  //Verifica e bloqueia o acionamento do botão
  if (bt == 1 && flag1 && !flag2){
    flag1 = false;
  }
  if (bt == 0 && !flag1 && !flag2){
    flag2 = true;
  }
```

```
  if (texto == "") {
    texto = reciveEth();
  }

  Serial.print(texto)
  //Verificação de finalização de comando
  if (texto == "Fim" && a0 == 1) {
    flag1 = true;
    flag2 = false;
    flag3 = false;
    count = 0;  //Zeragem do contador
  }

  /*
  Serial.print("\tcount: ");
  Serial.print(count);
  Serial.print("\tFlag: ");
  Serial.println(flag2);
  */

  if (flag2 && count < 2){
    //Condição para acionamento de avanço ou recuo do pistão quando o sensor for acionado.
    if (a0 == 1 && a1 == 0){
      sendEth(avanco);
      count++;  //Incremento do contador
      flag3 = true;
    }
    while (a1 != 1&&flag3) {
      sensor();
    }
    if (flag3) {
      tempo = millis();
    }
    flag3 = false;
    if(a0 == 0 && a1 == 1 && (millis() - tempo) >= 5000) {
      sendEth(recuo);
      flag1 = true;
      flag2 = false;
      count = 0;
    }
  }
  delay(100);
}

void sensor() {
  bt = digitalRead(pinbt); //Atribuição do valor da leitura do botão
  a0 = digitalRead(pina0); //Atribuição do valor da leitura do sensor A0
  a1 = digitalRead(pina1); //Atribuição do valor da leitura do sensor A1

  Serial.print("Botão: ");
  Serial.print(bt, DEC);
  Serial.print("\tSensor A0: ");
  Serial.print(a0, DEC);
  Serial.print("\tSensor A1: ");
  Serial.print(a1, DEC);
}

char reciveEth() {
  int packetSize = Udp.parsePacket(); //Definição do tamanho do pacote
  //Verifica se o pacote tem dados
  if (packetSize) {
    /*
    Udp.read(packetBuffer, MaxSize) - Leitura de pacotes
    packetBuffer - Buffer para capturar entradas de pacote
    MaxSize - Tamanho máximo do buffer
    */
    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
  }
  return packetBuffer;
}

void sendEth(char texto) {
  Udp.beginPacket(remoteip, port);  //Inicialização de pacote para envio da menssagem
  Udp.write(texto);  //Ecrita no pacote do texto
  Udp.endPacket();  //Finalização de pacote e envio da menssagem
}

void iniEthernet() {
  while (!Serial){
    ; //Espera pela conexão com a porta serial
  }
```

```
122   /*
123   Ethernet.begin(mac, ip, gateway, subnet) - Inicialização da conexão via Ethernet
124   mac - midia de controle de acesso de endereco da maquina (arduino/array 6 bytes)
125   ip - endereco de IP da maquina (arduino/array 4 bytes)
126   gateway - endereco de IP da porta network (array 4 bytes)
127   subnet - mascara da network (array 4 bytes)
128   */
129   Ethernet.begin(mac,ip);
130
131   //Verificação de que o Controlador de Ethernet está conectado
132   if (Ethernet.hardwareStatus() == EthernetNoHardware){
133   Serial.println("Controlador de Ethernet não encontrado.");
134     while (Ethernet.hardwareStatus() == EthernetNoHardware) {
135       delay(1000); //Atraso de 1 segundo
136     }
137   }
138   if (Ethernet.hardwareStatus() == EthernetW5100) {
139     Serial.println("Controlador de Ethernet W5100 detectado.");
140   }
141   else if (Ethernet.hardwareStatus() == EthernetW5200) {
142     Serial.println("Controlador de Ethernet W5200 detectado.");
143     flag1 = true;
144   }
145   else if (Ethernet.hardwareStatus() == EthernetW5500) {
146     Serial.println("Controlador de Ethernet W5500 detectado.");
147     flag1 = true;
148   }
149
150   //Teste do Status do Link (Ligado/Desligado)
151   if (Ethernet.linkStatus() == LinkOFF && flag1){
152     Serial.println("Status do Link = DESLIGADO");
153   }
154
155   flag1 = true;
156
157   Udp.begin(port);  //Inicializa o protocolo UDP na Ethernet
158 }
```

Receptor.ino:

Receptor

```
#include <SPI.h>  //Inclusão da biblioteca SPI.h
#include <Ethernet.h> //Inclusão da biblioteca Ethernet.h
#include <Udp.h>  //Inclusão da biblioteca Udp.h
#define port 8888 //Definição da porta de comunicação
#define pinrele 8  //Definição de uma constante do rele

EthernetUDP Udp;
byte mac[]= {0x90, 0xA2, 0xDA, 0x00, 0x64, 0x44}; //Atribuição do MAC Address do arduino recepitor
byte ip[] = {192,168,1,151};  //Atribuição do IP Address do arduino receptor
byte remoteip[] = {192,168,1,131};  //Atribuição do IP Address do arduino emissor

char packetBuffer[UDP_TX_PACKET_MAX_SIZE];  //Definição do tamanho do packetBuffer (24 byte)
char rele;  //Definição da variável rele
int count = 0;
volatile bool flag = false; //Definição de flag

void setup() {
  Serial.begin(9600); //Inicialização da porta Serial
  iniEthernet();  //Inicialização do adaptador de rede

  pinMode(pinrele,OUTPUT);  //Definição da pinagem do rele como saida
}

void loop() {
  rele = reciveEth(); //Atribuição do valor recebido via ethernet

  //Verificação de que foi recebido dados
  if (flag) {
    digitalWrite(pinrele,rele);  //Escrita no pino do rele (liga/desliga)
    flag = false;
      count++;
  }

  if (count >= 2) {
    sendEth("Fim");
  }

  delay(100);
}

char reciveEth() {
  int packetSize = Udp.parsePacket(); //Definição do tamanho do pacote
  //Verifica se o pacote tem dados
  if (packetSize) {
    Serial.println("Teste");
    flag = true; //Atribuição de recebimento de dado

    /*
    Udp.read(packetBuffer, MaxSize) - Leitura de pacotes
    packetBuffer - Buffer para capturar entradas de pacote
    MaxSize - Tamanho máximo do buffer
    */
    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);

    //val[1] = packetBuffer[0]; //Atribuição do dado recebido
  }
  return packetBuffer[0]; //Retorno dos valores da variável
}

void sendEth(char texto) {
```

```
  Udp.beginPacket(remoteip, port);  //Inicialização de pacote para envio da menssagem
  Udp.write(texto);  //Ecrita no pacote do texto
  Udp.endPacket();  //Finalização de pacote e envio da menssagem
}

void iniEthernet() {
 while (!Serial){
   ; //Espera pela conexão com a porta serial
 }

 /*
 Ethernet.begin(mac, ip, gateway, subnet) - Inicialização da conexão via Ethernet
 mac - midia de controle de acesso de endereco da maquina (arduino/array 6 bytes)
 ip - endereco de IP da maquina (arduino/array 4 bytes)
 gateway - endereco de IP da porta network (array 4 bytes)
 subnet - mascara da network (array 4 bytes)
 */
 Ethernet.begin(mac,ip);

 //Verificação de que o Controlador de Ethernet está conectado
 if (Ethernet.hardwareStatus() == EthernetNoHardware){
 Serial.println("Controlador de Ethernet não encontrado.");
   while (Ethernet.hardwareStatus() == EthernetNoHardware) {
     delay(1000); //Atraso de 1 segundo
   }
 }
 if (Ethernet.hardwareStatus() == EthernetW5100) {
   Serial.println("Controlador de Ethernet W5100 detectado.");
 }
 else if (Ethernet.hardwareStatus() == EthernetW5200) {
```

```
    Serial.println("Controlador de Ethernet W5200 detectado.");
    flag = true;
  }
  else if (Ethernet.hardwareStatus() == EthernetW5500) {
    Serial.println("Controlador de Ethernet W5500 detectado.");
    flag = true;
  }

  //Teste do Status do Link (Ligado/Desligado)
  if (Ethernet.linkStatus() == LinkOFF && flag){
    Serial.println("Status do Link = DESLIGADO");
  }

  flag = true;

  Udp.begin(port);  //Inicializa o protocolo UDP na Ethernet
}
```

3 Conclusão

3.1 Sequência indicada funcionando conforme especificado

O link a seguir é de um vídeo (no formato *Shorts*, do *Youtube*) onde está sendo demonstrado o funcionamento do pistão A, infelizmente não foi possível alcançar o funcionamento correto da sequência indicada para o grupo 1.

https://youtube.com/shorts/diiSFLgjuN4

A fim de facilitar a visualização dos códigos e também permitir uma fonte de auxílio para as próximas turmas de Redes Industriais com o professor Jean-Paul, o seguinte link redireciona para um repositório no site *Github* onde os códigos mostrados nesse relatório estão na íntegra e também outros códigos trabalhados na tentativa do sistema funcionar.

https://github.com/AlissonCV/Projeto_Redes.git