


```

const unsigned long PAUSE_TIME = 15000; // Intervalo de envio (15s)
unsigned long timeout; // Controle de tempo
int count = 0; // Contador de transmissões

// -----
// Protótipo da função de tratamento de eventos
void event_handler(Event);

// -----
void setup() {
    Serial.begin(115200);
    Serial.println(F("--- SMW_SX1276M0 P2P ---"));

    pinMode(LED_PIN, OUTPUT); // Configura LED como saída
    digitalWrite(LED_PIN, LOW); // LED começa apagado

    lorawan.setPinReset(5); // Define pino de reset do LoRa
    lorawan.reset(); // Reinicializa módulo LoRa

    LoRaSerial.begin(115200, SERIAL_8N1, RXD2, TXD2); // Inicia comunicação serial com LoRa

    lorawan.event_listener = &event_handler; // Define função de evento
    Serial.println(F("Handler configurado"));
}

// -----
void loop() {
    lorawan.listen(); // Mantém LoRa escutando mensagens

    if (timeout < millis()) { // Verifica tempo para próxima transmissão
        count++;
        if (count > 255) count = 0; // Evita estouro de variável

        char data[] = "SENHA"; // Dados a serem enviados

        Serial.println("=====");
        Serial.print(F("Mensagem Enviada (P/ NÓ B): "));
        Serial.println(data);
        Serial.println("=====");

        response = lorawan.sendT(10, data); // Envia mensagem para a porta 10
        timeout = millis() + PAUSE_TIME; // Atualiza o tempo de envio
    }
}

// -----
// Função de tratamento de eventos LoRa
void event_handler(Event type) {
    if (type == Event::JOINED) {
        Serial.println(F("Conectado"));
    }
    else if (type == Event::RECEIVED) { // Mensagem recebida
        Serial.println(F("Mensagem de texto recebida"));
        delay(50);
        lorawan.flush();

        uint8_t port;
        Buffer buffer;
        response = lorawan.readT(port, buffer); // Lê dados recebidos
        if (response == CommandResponse::OK) {
            String mensagemRecebida = "";
            while (buffer.available()) {
                mensagemRecebida += (char)buffer.read(); // Converte dados para string
            }
            Serial.print(F("Mensagem: "));
            Serial.println(mensagemRecebida);
            Serial.print(F(" na porta "));
            Serial.println(port);

            // Se a mensagem for "ACESSO LIBERADO!", acende o LED
            if (mensagemRecebida.equals("ACESSO LIBERADO!")) {
                Serial.println(F("Recebido: ACESSO LIBERADO! - Ligando LED."));
                digitalWrite(LED_PIN, HIGH);
            }
        }
        else if (type == Event::RECEIVED_X) { // Mensagem recebida em hexadecimal
            Serial.println(F("!Mensagem Hexadecimal recebida e convertida em texto!"));
            delay(50);
            lorawan.flush();

            uint8_t port;

```

```

Buffer buffer;
response = lorawan.readX(port, buffer); // Lê dados hexadecimais
if (response == CommandResponse::OK) {
    Serial.print(F("Mensagem Recebida (NÓ B): "));
    String mensagemHex = "";

    while (buffer.available()) {
        uint8_t byteHex = buffer.read();
        mensagemHex += (char)byteHex; // Converte para string
    }

    String mensagemConvertida = "";
    for (int i = 0; i < mensagemHex.length(); i += 2) {
        String hexByte = mensagemHex.substring(i, i + 2);
        char caractere = (char)strtol(hexByte.c_str(), NULL, 16);
        mensagemConvertida += caractere; // Converte HEX para ASCII
    }

    Serial.print(mensagemConvertida);
    Serial.print(F(" | Na porta: "));
    Serial.println(port);

    // Se a mensagem for "ACESSO LIBERADO!", acende e pisca o LED
    if (mensagemConvertida.equals("ACESSO LIBERADO!")) {
        Serial.println(F("Recebido: ACESSO LIBERADO! - Ligando LED.));
        digitalWrite(LED_PIN, HIGH);
        delay(1000);
        digitalWrite(LED_PIN, LOW);
        delay(1000);
    }
}
}
}

```

2. Receptor: (30/01/25):

C/C++

```

#if !defined(ARDUINO_ESP32_DEV) // Verifica se o ESP32 está sendo usado
#error Use este exemplo com o ESP32
#endif

// -----
// Bibliotecas

#include <RoboCore_SMW_SX1276M0.h> // Biblioteca do módulo LoRa
#include <string.h> // Biblioteca para manipulação de strings

// -----
// Variáveis
#include <HardwareSerial.h> // Biblioteca para comunicação serial por hardware
HardwareSerial LoRaSerial(2); // Define o canal serial 2 para comunicação LoRa
#define RXD2 16 // Define o pino RX para a comunicação serial LoRa
#define TXD2 17 // Define o pino TX para a comunicação serial LoRa

SMW_SX1276M0 lorawan(LoRaSerial); // Cria o objeto LoRaWAN para o módulo SMW_SX1276M0

CommandResponse response; // Variável para armazenar a resposta dos comandos LoRa

// Endereços e chaves para configuração do dispositivo e comunicação
const char DEVADDR[] = "00000002"; // Endereço do dispositivo atual
const char DEVADDR_P2P[] = "00000001"; // Endereço de destino para comunicação P2P
//const char APPSKEY[] = "0000000000000000000000000000000c"; // Chave de sessão do aplicativo
//const char NWKSKEY[] = "0000000000000000000000000000000b"; // Chave de sessão da rede

const unsigned long PAUSE_TIME = 3000; // Pausa de 30 segundos entre transmissões (em ms)
unsigned long timeout; // Variável para controle de tempo
int count = 0; // Contador para número de transmissões

// -----
// Protótipos
void event_handler(Event); // Função para gerenciar eventos LoRa
// -----

```

```

// -----

void setup() {
    // Inicializa a UART para depuração
    Serial.begin(115200);
    Serial.println(F("--- SMW_SX1276M0 P2P ---"));

    // Definição do pino de reset do módulo
    lorawan.setPinReset(5);
    lorawan.reset(); // Realiza um reset no módulo

    // Inicializa a UART para o módulo LoRa
    LoRaSerial.begin(115200, SERIAL_8N1, RXD2, TXD2);

    // Define a função de tratamento de eventos
    lorawan.event_listener = &event_handler;
    Serial.println(F("Handler configurado"));
}
// -----
// -----

void loop() {
    // Escuta dados recebidos do módulo
    lorawan.listen();
}
// -----
// -----

// Função para tratar os eventos do módulo
// @param (type) : o tipo do evento [Event]
void event_handler(Event type){
    // Lê a mensagem
    uint8_t port;
    Buffer buffer;
    char lido;
    // Verifica se o evento é de conexão
    if(type == Event::JOINED){
        Serial.println(F("Conectado"));
    }
    // Verifica se um texto foi recebido
    else if(type == Event::RECEIVED){
        Serial.println(F("Mensagem de texto recebida"));

        // Aguarda um tempo para limpar dados restantes do evento
        delay(50);
        lorawan.flush();

        response = lorawan.readT(port, buffer);
        if(response == CommandResponse::OK){
            Serial.print(F("Mensagem: "));
            while(buffer.available()){
                //lido = buffer.read();
                //Serial.println(lido);
                //Serial.println('\n');
            }
            Serial.print(F("na porta:"));
            //Serial.println(port);
        }
    }
    else if (type == Event::RECEIVED_X) {
        delay(50);
        lorawan.flush();

        uint8_t port;
        Buffer buffer;
        response = lorawan.readX(port, buffer);
        if (response == CommandResponse::OK) {
            String mensagemHex = "";
            while (buffer.available()) {
                uint8_t byteHex = buffer.read();
                mensagemHex += (char)byteHex;
            }

            String mensagemConvertida = "";
            for (int i = 0; i < mensagemHex.length(); i += 2) {
                String hexByte = mensagemHex.substring(i, i + 2);
                char caractere = (char)strtol(hexByte.c_str(), NULL, 16);
                mensagemConvertida += caractere;
            }

            Serial.println("=====");
            Serial.print(F("Mensagem Recebida (Do NÓ A): "));

```

```

Serial.print(mensagemConvertida);
Serial.print(F(" | Na porta: "));
Serial.println(port);
Serial.println("=====");

if (mensagemConvertida.equals("SENHA")) {
  Serial.println(F("!Mensagem Hexadecimal recebida e convertida em texto! Enviando resposta para A..."));
  char resposta[] = "ACESSO LIBERADO!";
  response = lorawan.sendT(port, resposta);

  if (response == CommandResponse::OK) {
    Serial.println(F("Resposta enviada com sucesso!"));
  } else {
    Serial.println(F("Erro ao enviar a resposta!"));
  }
}
}
}

```

Fluxo Geral da Comunicação LoRa P2P:

- ① Início
- ② Configuração do ESP32 e do módulo LoRa em ambos os dispositivos
- ③ O Nó A envia a mensagem "SENHA" para o Nó B
- ④ O Nó B recebe a mensagem e verifica o conteúdo
 - ♦ Se a mensagem for "SENHA", ele responde enviando para Nó A: "ACESSO LIBERADO!"
- ⑤ O Nó A recebe a resposta do Nó B
 - ♦ Se a resposta for "ACESSO LIBERADO!", o LED é aceso
- ⑥ O processo se repete continuamente após um intervalo de tempo

Isso cria um sistema simples de solicitação e autorização entre os dois dispositivos via LoRa.

The image displays two side-by-side screenshots of the Arduino IDE, showing the code for a LoRa P2P communication system. The left window is titled 'transmissor_copy_20250131192140 | Arduino IDE 2.3.3' and the right window is titled 'receptor_copy_20250131192208 | Arduino IDE 2.3.3'. Both windows show the code for an ESP32 DevKit V1.

Transmitter Code (Left Window):

```

// Se a mensagem for "ACESSO LIBERADO!", acende e pisca o LED
if (mensagemConvertida.equals("ACESSO LIBERADO!")) {
  Serial.println(F("Recebido: ACESSO LIBERADO! - Ligando LED."));
  digitalWrite(LED_PIN, HIGH);
  delay(1000);
  digitalWrite(LED_PIN, LOW);
  delay(1000);
}
}
}

```

Receiver Code (Right Window):

```

String mensagemConvertida = "";
for (int i = 0; i < mensagemHex.length(); i += 2) {
  String hexByte = mensagemHex.substring(i, i + 2);
  char caractere = (char)strtol(hexByte.c_str(), NULL, 16);
  mensagemConvertida += caractere;
}

Serial.println("Mensagem Recebida (Do Nó A): ");
Serial.print(mensagemConvertida);
Serial.print(F(" | Na porta: "));
Serial.println(port);
Serial.println("=====");

if (mensagemConvertida.equals("SENHA")) {
  Serial.println(F("!Mensagem Hexadecimal recebida e convertida em texto! Enviando resposta para A..."));
  char resposta[] = "ACESSO LIBERADO!";
  response = lorawan.sendT(port, resposta);

  if (response == CommandResponse::OK) {
    Serial.println(F("Resposta enviada com sucesso!"));
  } else {
    Serial.println(F("Erro ao enviar a resposta!"));
  }
}
}
}

```

Serial Monitor Output (Left Window):

```

08:29:13.346 -> =====
08:29:13.388 -> Mensagem Enviada (P/ Nó B): SENHA
08:29:13.388 -> Mensagem Enviada (P/ Nó B): SENHA
08:29:16.835 -> !Mensagem Hexadecimal recebida e convertida em texto!
08:29:17.190 -> Mensagem Recebida (Nó B): ACESSO LIBERADO! | Na porta: 10
08:29:17.190 -> Recebido: ACESSO LIBERADO! - Ligando LED.

```

Serial Monitor Output (Right Window):

```

08:29:15.122 -> =====
08:29:15.122 -> Mensagem Recebida (Do Nó A): SENHA | Na porta: 10
08:29:15.122 -> Mensagem Recebida (Do Nó A): SENHA | Na porta: 10
08:29:15.122 -> !Mensagem Hexadecimal recebida e convertida em texto! Enviando resposta para A...
08:29:16.123 -> Resposta enviada com sucesso!

```

- Limitações que surgem com essa estratégia
 1. A segurança do sistema pode ser comprometida pois a chave de confirmação é pouco pode ser vazada ou perdida
 2. Se a mensagem não for entregue corretamente em ambas as direções não existe uma maneira de forçar o envio e o fluxo vai continuar sem a entrega da palavra de acesso.

Esses problemas podem ser superados com uma solução mais completa no futuro.