Código LoRa Sender:

```
/*LoRa sender*/

#include<SPI.h>/*comunicação serial*/
#include<LoRa.h>/*comunicação wifi*/
#include<Wire.h>/*comunicação i2c*/
#include "SSD1306.h"/*comunicação com o display*/

/*Definição dos pinos*/
#define SCK 5
#define MISO 19
#define SS 18
#define RST 14
#define DI00 26 /*Interrompe a requisição*/
#define BAND 915E6 /*Frequencia de radio - Podemos usar também a 433E6 e a 868E6*/
#define PABOOST true

unsigned int counter = 0;

SSD1306 display(0x3c,4,15);/*Construtor do objeto que controla o display*/

String rssi = "RSSI --"
String packSize = "--"
String packet;

void setup() {

// Configura os pinos de saida

pinMode(16, OUTPUT);
pinMode(25,OUTPUT);

digitalWrite(16,LOW);//Reset
delay(50);
digitalWrite(16,HIGH);

display.init(); //inicializa o display

SPI.begin(SCK,MISO,MOSSI,SS);// Inicia a comunicação
```

```
LoRa.setPins(SS,RST,DI00);

/*Verificando se o display iniciou corretamente*/

if(!LoRa.begin(BAND,PABOOST)){
  display.drawString(0,0,"Erro ao iniciar o LoRa");
  display.display();//Mostra o conteúdo na tela
while(1);
}
display.drawString(0,0,"LoRa iniciado com sucesso!");
display.display();
delay(1000);

}

void loop() {

/*Apaga o conteúdo do display*/

display.clear();
display.drawString(0,0,"Enviando")
display.drawString(40,26,String(counter));
display.display();

LoRa.beginPacket();
LoRa.print("Hello World!");
LoRa.print(counter);
LoRa.endPacket();

counter++;

digitalWrite(25, HIGH);// Liga o LED indicativo
delay(500);
digitalWrite(25,LOW);//Desliga o LED indicativo
delay(500);

}
```

Código LoRa Receiver:

```
/*LoRa sender*/

#include<SPI.h>/*comunicação serial*/
#include<LoRa.h>/*comunicação wifi*/
#include<Wire.h>/*comunicação i2c*/
#include "SSD1306.h"/*comunicação com o display*/

/*Definição dos pinos*/
#define SCK 5
#define MISO 19
#define SS 18
#define RST 14
#define DI00 26 /*Interrompe a requisição*/
#define BAND 915E6 /*Frequencia de radio - Podemos usar também a 433E6 e a 868E6*/
#define PABOOST true

unsigned int counter = 0;

SSD1306 display(0x3c,4,15);/*Construtor do objeto que controla o display*/

String rssi = "RSSI --"
String packSize = "--"
String packet;

void setup() {

// Configura os pinos de saida

pinMode(16, OUTPUT);
pinMode(25,OUTPUT);

digitalWrite(16,LOW);//Reset
delay(50);
digitalWrite(16,HIGH);

display.init(); //inicializa o display
```

```
SPI.begin(SCK,MISO,MOSSI,SS);// Inicia a comunicação
LoRa.setPins(SS,RST,DI00);

/*Verificando se o display iniciou corretamente*/

if(!LoRa.begin(BAND,PABOOST)){
  display.drawString(0,0,"Erro ao iniciar o LoRa");
  display.display();//Mostra o conteúdo na tela
while(1);
}
display.drawString(0,0,"LoRa iniciado com sucesso!");
display.drawString(0,10,"Aguardando dados...")
display.display();
delay(1000);

LoRa.receive();
}

void loop() {

int pacaketSize = LoRa.parsePacket();

if(packetSize){

  cbk(packetSize);// Função que recupera o tamanho do pacote do contador
  delay(10);
}

}
void loraData(){

 display.clear();
 display.setTextAlignment(TEXT_ALIGN_LEFT);
 display.setFont(ArialMT_Plain_16);
 displaydrawString(0,18, "RX" + packetSize + "Bytes");
 display.drawStringMaxWidth(0,39,128, packet);
 display.drawString(0,0,rssi);
 display.display();
}
```

Código LoRa Send and Receive:

Master:

```
#ifdef MASTER

#define INTERVAL 500

long lastSendTime = 0;

void receive(){

  int packetSize = LoRa.parsePacket();

    if(packetSize < SETDATA.length()){
      String received = "";

        while(LoRa.available()){
          received+=(char)LoRa.read();

        }

      int index = received.indexOf(SETDATA);
        if(index >= 0){

          String data = received.subString(SETDATA.length());
          String waiting = String(milis() - lastSendTime);

          display.clear();
          display.drawString(0,0,"Recebeu" + data);
          display.drawString(0,10,"Tempo" + waiting + "ms");
          display.display();
        }
      }
}

void loop() {

    if(milis() - lastSendTime > INTERVAL){
      lastSendTime = milis();
```

```
 send();
   }
  receive();
}
#endif
```

Slave:

```
#ifdef MASTER

 int count = 0;

 void setup(){
  Serial.begin(115200);

  setupDisplay();
  setupLoRa();
  display.clear();
  display.drawString(0,0,"Slave esperando");
  display.display();

 }

 void loop(){
   int packetSize = LoRa.parsePacket();

   if(packetSize == GETDATA.length()){

     String received = "";

      while(LoRa.available()){

        received += (char)LoRa.read();

      }

      if(received.equals(GETDATA)){
        String data = readData();
```

```
        Serial.println("Criando pacote para envio");

        LoRa.beginPacket();
        LoRa.print(SETDATA + data);

        LoRa.endPacket();

        display.clear();
        display.drawString(0,0, "Enviou" + String(data));
        display.display();
      }

    }

}
String readData(){
  return Strign(count++);
}
#endif
```

Corpo do código:

```
/*LoRa Send and Receive*/

#include<SPI.h>/*comunicação serial*/
#include<LoRa.h>/*comunicação wifi*/
#include<Wire.h>/*comunicação i2c*/
#include "SSD1306.h"/*comunicação com o display*/

#define MASTER
/*Definição dos pinos*/
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DI00 26 /*Interrompe a requisição*/

#define BAND 433E6 /*Frequencia de radio - Podemos usar também a 915E6 e a 868E6*/
```

```
const String GETDATA = "getData"
const String SETDATA = "setData"

SSD1306(0X3C,4,15);

void setupDisplay(){
  pinMode(16, OUTPUT);
  digitalWrite(16,LOW);
  digitalWrite(16,HIGH);

  display.init();
  display.flipScreenVertically();
  display.setFont(Arial_Plain_10);
  display.setTextAligment(TEXT_ALIGN_LEFT);


}

void setupLoRa() {

  SPI.begin(SCK,MISO,MOSI,SS);
  LoRa.setPins(SS,RST,DI00);

  if(!LoRa.begin(BAND,true)){
    display.clear();
    display.drawString(0,0,"Erro ao iniciar o LoRa");
    display.display();
    while(1);
  }
  LoRa.enableCrc();

  LoRa.receive();

}

void setup(){
  Serial.begin(115200);
  setupDisplay();
  setupLoRa();

  display.clear();
```

```
  display.drawString(0,0,"Master");
  display.display();
}


void loop() {

    if(milis() - lastSendTime > INTERVAL){
      lastSendTime = milis();
      send();
    }
  receive();
}
```