TRUYỀN NHẬN DỮ LIỆU BẰNG LORA

1. Tổng quan về LORA:

LoRa là viết tắt của Long Range được phát triển bởi Semtech, là một kỹ thuật điều chế có nguồn gốc từ kỹ thuật "Chirp Spread Spectrum" (CSS), trong đó Chirp là viết tắt của "Compressed High Intetsity Radar Pulse", là một kỹ thuật sử dụng rất phổ biến trong sonar và radar. Tín hiệu Chirp có biên độ không đổi và tần số thay đổi, nếu tần số thay đổi từ tần số thấp nhất đến tần số cao nhất thì được gọi là up-chirp và thay đổi từ tần số cao nhất đến thấp nhất được gọi là down-chirp.

2. Nguyên lý hoạt động:

LoRa dựa trên kỹ thuật điều chế Chirp Spread Spectrum. Khi các dữ liệu được tạo xung với tần số cao để tạo ra những tín hiệu có dải tần cao hơn. Các tín hiệu này sẽ được mã hóa theo các chuỗi chirp signal (tín hiệu hình sin thay đổi theo thời gian) trước khi được gửi đi từ anten.

Có hai loại chirp signal, bao gồm:

- Tần số up-chirp tăng theo thời gian.
- Tần suất của Down-chirp giảm dần theo thời gian

Loại kỹ thuật này sẽ sử dụng các xung tần số cao để chia nhỏ dữ liệu nhằm tạo ra tín hiệu có dải tần cao hơn dải tần của dữ liệu gốc. Sau đó tín hiệu cao tần sẽ tiếp tục được mã hóa theo chuỗi tín hiệu chirp rồi truyền đến anten để truyền đi.

Nguyên tắc hoạt động này hỗ trợ thiết bị giảm độ phức tạp và tăng độ chính xác cần thiết cho mạch nhận để có thể giải mã và điều chỉnh lại dữ liệu. LoRa không yêu cầu nhiều công suất phát mà vẫn có thể truyền đi xa, vì tín hiệu LoRa có thể nhận được ở khoảng cách xa ngay cả khi cường độ tín hiệu thấp hơn nhiễu xung quanh.

Băng tần hoạt động của công nghệ LoRa nằm trong khoảng từ 430MHz đến 915MHz, áp dụng cho các khu vực khác nhau trên thế giới cụ thể

- Dải băng tần 430MHz cho khu vực châu Á
- Dải băng tần 780MHz cho khu vực Trung Quốc
- Dải băng tần 433MHz hoặc 866MHz cho khu vực châu Âu
- Dải băng tần 915MHz cho khu vực USA

3. Các thông số cơ bản:

BandWitdh-BW xác định biên độ tần số mà tín hiệu chirp có thể thay đổi. Các chip LoRa khác nhau sẽ cho phép tùy biến cấu hình các mức băng thông khác nhau, nhưng thông thường sẽ cấu hình ba mức băng thông phổ biến là 125kHz, 250kHz và 500 kHz.

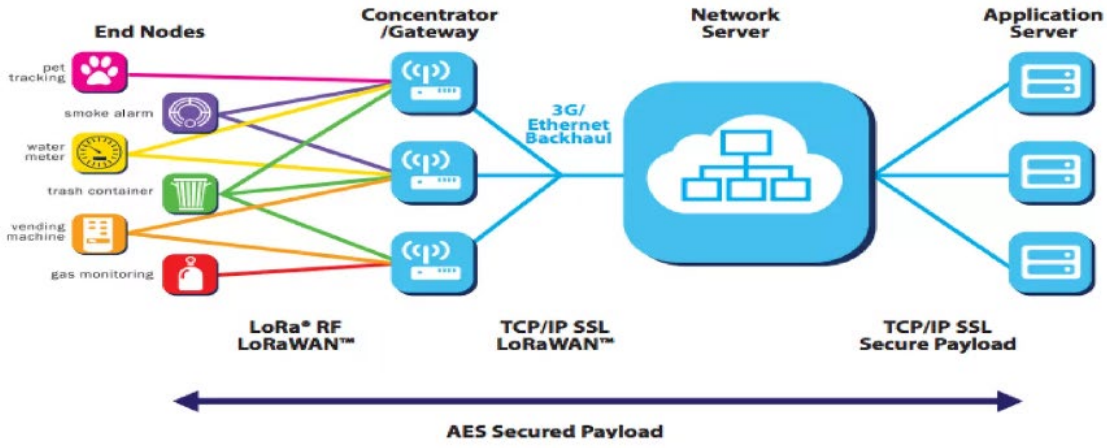
Băng thông cao sẽ cho phép mã hóa tín hiệu nhanh hơn, giúp thời gian truyền dữ liệu nhanh hơn nhưng bù lại khoảng cách truyền cũng sẽ ngắn đi.

Spreading Factor - SF xác định số lượng tín hiệu chirp khi mã hóa tín hiệu đã được điều chế tần số (chipped signal) của dữ liệu, SF là các giá trị nguyên từ 7 đến 12. Ví dụ nếu SF = 12 có nghĩa là 1 mức logic của tín hiệu sẽ được mã hóa bởi 12 xung tín hiệu chirp. Giá trị SF càng lớn thì thời gian truyền dữ liệu sẽ lâu hơn nhưng đổi lại tỉ lệ lỗi bit BER sẽ giảm và khoảng cách truyền cũng sẽ xa hơn.

Coding Rate - CR là số lượng bit được tự thêm vào trong payload gói tin LoRa để mạch nhận có thể sử dụng để phục hồi lại một số bit dữ liệu đã nhận sai và từ đó phục hồi được nguyên vẹn dữ liệu payload. CR là các giá trị nguyên từ 1 đến 4 và thường biểu thị ở dạng **4/CR+4** (ví dụ: 4/5, 4/6, 4/7, 4/8). Do đó, sử dụng CR càng cao thì khả năng nhận dữ liệu đúng càng tăng, nhưng bù lại chip LoRa sẽ phải gửi nhiều dữ liệu hơn và làm tăng thời gian truyền.

4. LORAWAN network:

LORAWan là chuẩn giao tiếp dựa trên nền tảng công nghệ LORA và được định nghĩa và phát triển bởi tổ chức LORA Alliance. Ở mỗi vùng khác nhau trên thế giới thì thiết bị LORAWan phải cấu hình cho chip LORA hoạt động ở dãy băng tần cho phép như 433Mhz, 915MHz,…



Do đó trong 1 mạng LORAWan sẽ có 2 loại thiết bị:

- Device node: là các thiết bị cảm biến, hoặc các thiết bị giám sát được lắp đặt tại các vị trí làm việc ở xa để lấy và gửi dữ liệu về các thiết bị trung tâm.
- Gateway: là các thiết bị trung tâm sẽ thu thập dữ liệu từ các device node và gửi lên 1 server trung tâm để xử lý dữ liệu. Các thiết bị Gateway thường sẽ được đặt tại 1 vị trí có nguồn cung cấp và có các kết nối network như Wifi, LAN, GSM để có thể gửi dữ liệu lên server

5. Thực hành cấu hình mạng Lora

5.1. Module sử dụng: Moudle LORA E32 433T20D EBYTE và AS32-TTL-100

1. Moudle E32 433T20D:

Mạch thu phát EBYTE E32-433T20D sử dụng chip SX1278 của nhà sản xuất SEMTECH chuẩn giao tiếp LORA, chuẩn LORA mang đến hai yếu tố quan trọng là tiết kiệm năng lượng và khoảng cách phát siêu xa (Ultimate long range wireless solution), ngoài ra nó còn có khả năng cấu hình để tạo thành mạng nên hiện tại được phát triển và sử dụng rất nhiều trong các nghiên cứu về IoT.

Mạch thu phát EBYTE E32-433T20D được tích hợp phần chuyển đổi giao tiếp SPI của SX1278 sang UART giúp việc giao tiếp và sử dụng rất dễ dàng, chỉ cần kết nối với Software của hãng để cấu hình địa chỉ , tốc độ và công suất truyền là có thể sử dụng

Thông số kỹ thuật:

- Model: EBYTE E32-433T20D Lora SX1278 433Mhz
- IC chính: SX1278 từ SEMTECH.
- Điện áp hoạt đông: 2.3 - 5.5 VDC
- Điện áp giao tiếp: TTL-3.3V
- Giao tiếp UART Data bits 8, Stop bits 1, Parity none, tốc độ từ 1200 - 115200.
- Tần số: 410 - 441Mhz
- Công suất: 20dbm (100mW)

- Khoảng cách truyền tối đa trong điều kiện lý tưởng: 3000m
- Tốc độ truyền: 0.3 - 19.2 Kbps (mặc định 2.4 Kbps)
- 512bytes bộ đệm.
- Hỗ trợ 65536 địa chỉ cấu hình.
- Kích thước: 21x36mm.

5.2. AS32-TTL-100:

Không khác gì so với E32 cả, chỉ khác ở chỗ 2 Mạch này được phát triển 2 nhà phát hành khác nhau E32 là của Ebyte còn AS32 là Ashining, và cả 2 có cách cấu hình LORA khác nhau

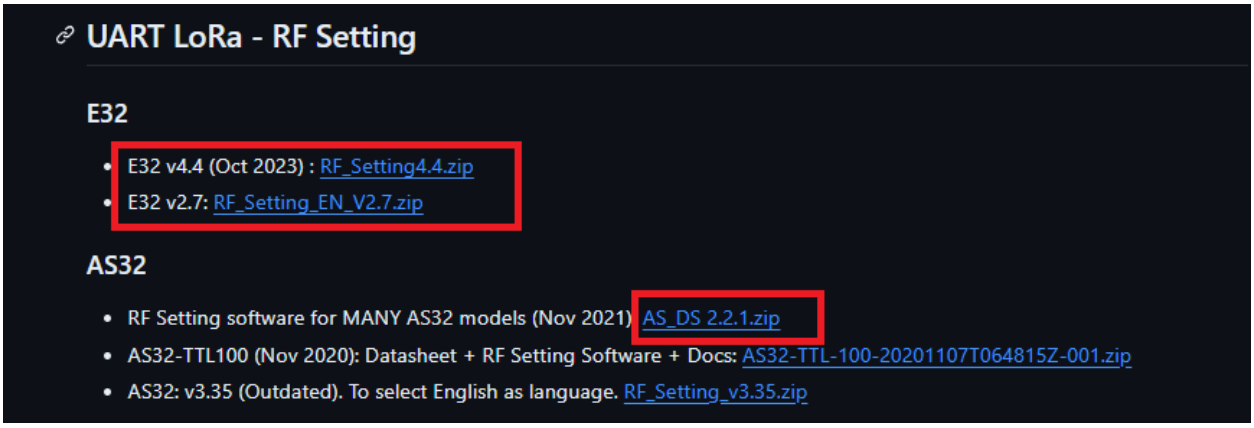
Thông số kỹ thuật:

- Model: AS32-TTL-100 RF
- IC chính: SX1278 từ SEMTECH.
- Điện áp hoạt đông: 2.3 – 5.5 VDC
- Điện áp giao tiếp: TTL
- Giao tiếp UART Data bits 8, Stop bits 1, Parity none, tốc độ từ 1200 – 115200.
- Tần số: 410 – 441Mhz
- Công suất: 20dbm (100mW)
- Khoảng cách truyền tối đa trong điều kiện lý tưởng: 3000m
- Tốc độ truyền: 0.3 – 19.2 Kbps (mặc định 2.4 Kbps)
- 512bytes bộ đệm.
- Hỗ trợ 65536 địa chỉ cấu hình.
- Kích thước: 21x36mm.

Bước 1: Cấu hình thông số cho Lora

Phần mềm để cấu hình cho Lora E32 và AS32:
https://github.com/IoTThinks/EasyLoRa/wiki/Download#uart-lora---rf-setting

Tải cái khung đỏ nếu tuỳ theo Lora đang cầm thuộc loại gì với E32 mình đang cấu hình sẽ là phiên bản 4.4



Cấu hình cho E32 như sau:

Để cấu hình cho Lora E32 trước tiên cần thiết bị USB TTL dòng chip CP2102 để giao tiếp với nó cách cắm cũng không khác gì so với làm Zigbee:

Lora E32	USB TTL
Tx	Rx
Rx	Tx
VCC	5v hoặc 3v3
GND	GND

Cắm vào máy táy rồi mở phần mềm lên giao diện của nó trông thế này: Có thể chọn ngôn ngữ Tiếng anh ở góc trên bên phải.

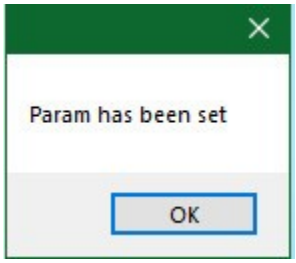

RF Setting V4.4
EBYTE
成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.
中文
English
COM3
OpenPort
Models
GetParam
SetParam
Preset
ParaSave
FileSet
Select File
UartRate
Parity
AirRate
Power
FEC
Fixed mode
WOR timing
IO mode
Address
Channel
Copyright@ Chengdu EByte Electronic Technology Co.Ltd
WebSite：www.ebyte.com

Sau đấy chọn Cổng COM USB TTL đã cắm và ấn OpenPort

Sau đó ấn GetParam nếu giao tiếp thành công thì nó sẽ hiển thị các thông số như thế này

Để cấu hình cho 2 LORA giao tiếp được với nhau thì chỉ cần chú ý về tốc độ băng thông tức là UartRate phải cùng với nhau, địa chỉ và kênh cũng phải trùng nhau, còn lại thì có thể để mặc định hết. Ví dụ trong hình mình để địa chỉ là 0 và kênh 15 thì các Lora muốn giao tiếp mạng này phải để cùng địa chỉ là 0 và kênh 15.

Sau khi cấu hình xong thì ấn SetParam đợi nó thông báo thành công thì kết thúc phần cài đặt cấu hình cho LORA dòng E32

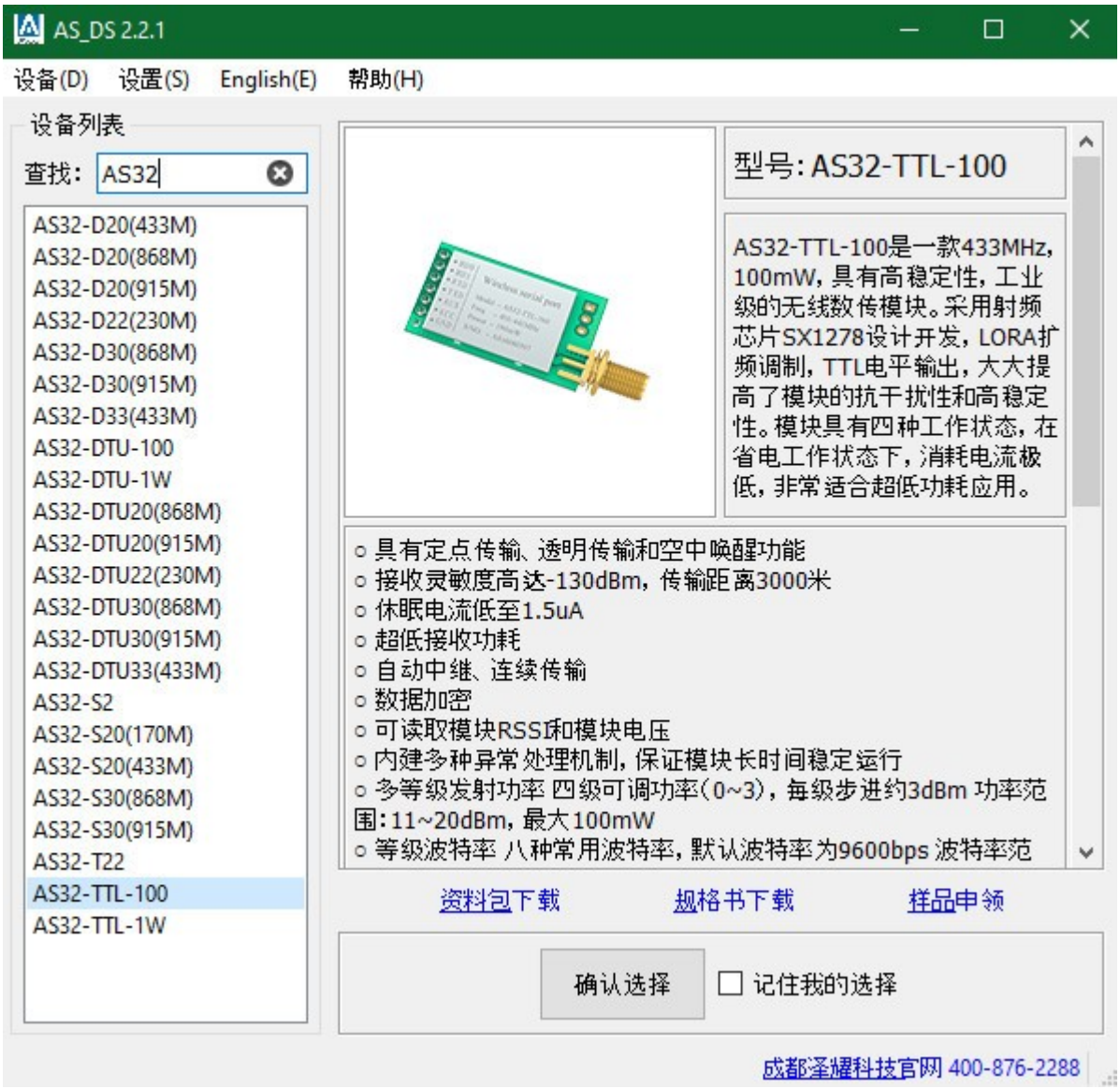


Cấu hình Lora AS32:

Loại dòng hơi phức tạp vì nó không hỗ trợ phiên bản tiếng anh tuy nhiên cách thức của nó vẫn giống dòng với E32

Đầu tiên vẫn dùng USB TTL cách thức cắm vẫn giống y như Lora E32 (Xem ở trên)

Sau khi cắm dây xong thì cắm vào máy tính xong mở phần mềm lên giao diện trông thế này. Sau đó gõ chữ AS32 giống trông hình và Click đúp vào loại AS32-TTL-100



Chọn Cổng COM (Chổ COM3 giống trong hình) sau đấy ấn vào cái khung đỏ để cài đặt thông số cho LORA.

Nếu nó hiện thế này thì có nghĩa là thành công nhận được LORA AS32

Điều chỉnh địa chỉ và kênh cho Lora AS32, để 2 Lora giao tiếp được với nhau thì địa chỉ và kênh của 2 lora phải cùng với nhau ví dụ trong hình mình đang cài đặt Lora ở địa chỉ 0 và kênh 15 thì Lora thứ 2 cũng phải là địa chỉ 0 kênh 15.

Điều chỉnh thông số xong thì ấn Cài đặt thông số LORA để ghi cấu hình và kết thúc bước cấu hình LORA.

Như vậy là kết thúc bước cấu hình mạng Lora, cấu hình Lora đơn giản rất nhiều so với loại Zigbee.

Lưu ý: Hiện tại mình chưa chuyển dữ liệu được LORA giữa 2 loại hãng khác nhau. Nghĩa là để giao tiếp các LORA, nếu bạn đang dùng E32 của Ebyte thì bạn chỉ giao tiếp được các dòng cùng hãng là Ebyte ví dụ E32 giao tiếp E32. Nếu bạn dùng AS32 của Ashining thì chỉ giao tiếp cùng hãng Ashining, ví dụ AS32 giao tiếp các dòng AS32.

6. Thực hành truyền nhận dữ liệu giữa các node trong mạng Lora

Trước tiên chúng ta phải tìm hiểu về Mode(Chế độ) có trong các chân của Lora AS32 và E32. Hai chân M0 và M1, M ở đây là Mode(Chế độ) 2 chân này đầu ra chỉ có 2 mức tín hiệu cao hoặc thấp nên ta có tân 4 chế độ cho Lora với mức 0 là mức LOW và 1 là mức HIGH

- Mode 0: M0(0) và M1(0) là chế độ Normal (Binh thường): Lora giao tiếp UART với các kênh truyền Lora không dây hoạt động. Quá trình truyện nhận dữ liệu hoạt động. Lora ở chế độ có thể truyền và nhận các Lora chế độ 1 và 0
- Mode 1: M0(0) và M1(1) là chế độ Wake-up (Thức dậy): Khác với Mode 0, ở Mode 1 có thêm 1 đoạn mã hóa tự động ban đầu trước khi dữ liệu được truyền đi nên nó có thể thông báo bên nhận làm việc ở chế độ 2. Nói dễ hiểu thì Lora ở chế độ Mode 1 có thể đánh thức các Lora đang ở chế độ 2. Lora chế độ 1 có thể truyền và nhận các Lora chế độ 0 và 1, có thể truyền cho Lora ở chế độ 2 nhưng không thể nhận từ Lora chế độ 2:
- Mode 2: M0(1) và M1(0) là chế độ Power Saving (Tiết kiệm năng lượng): Lora chế độ 2 này chỉ có thể nhận dữ liệu từ Lora ở chế độ 1, không thể truyền cho Lora bất kì chế độ nào
- Mode 3: M0(1) và M1(1) là chế độ Sleep (Ngủ): Là chế độ Lora không giao tiếp tất cả Lora kể cả nó nằm trong vùng mạng có thể giao tiếp

Test truyền nhận các Lora, cách cắm dây Lora AS32 và E32 vào Arduino Uno sau:

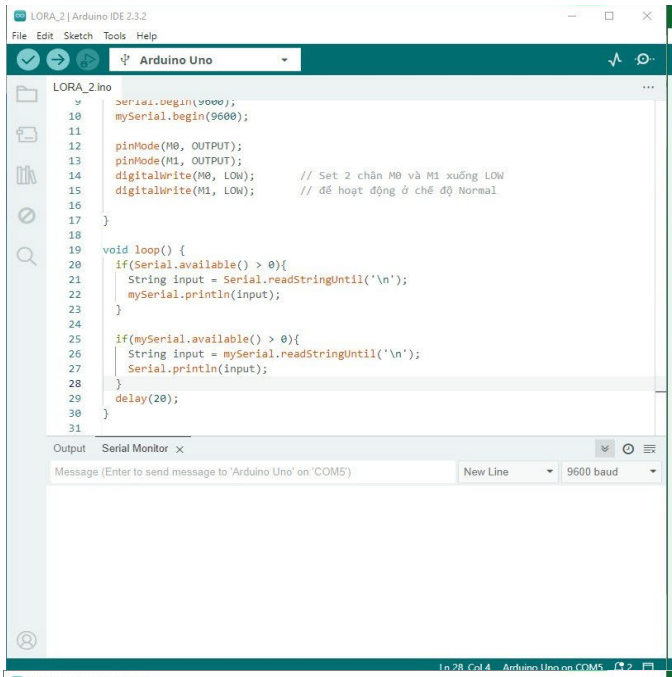
E32 hoặc AS32	Arduino Uno
M0	D8
M1	D9
VCC	5V
GND	GND
Tx	D6
Rx	D7

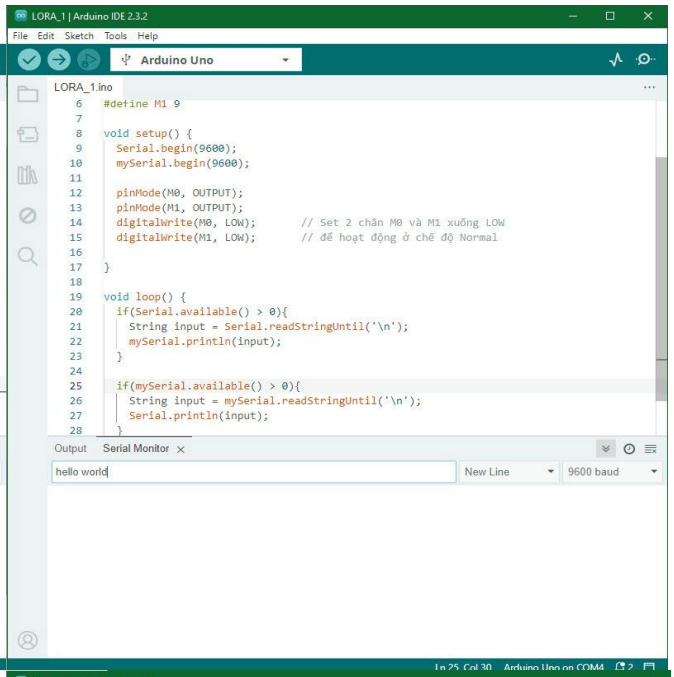
Đây là Code truyền nhận giữa 2 Lora sau, cả code truyền và nhận đều dùng chung 1 code:

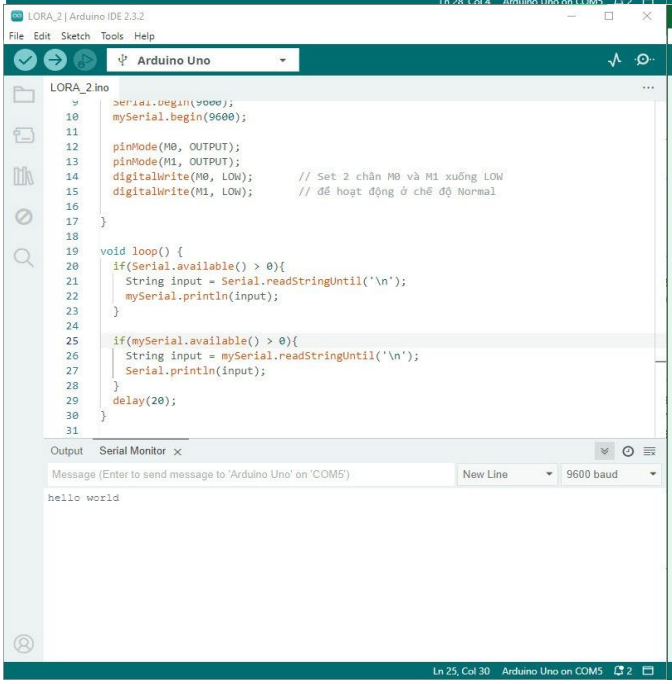
```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(6, 7); //TX, RX

#define M0 8
#define M1 9

void setup() {
  Serial.begin(9600);
```

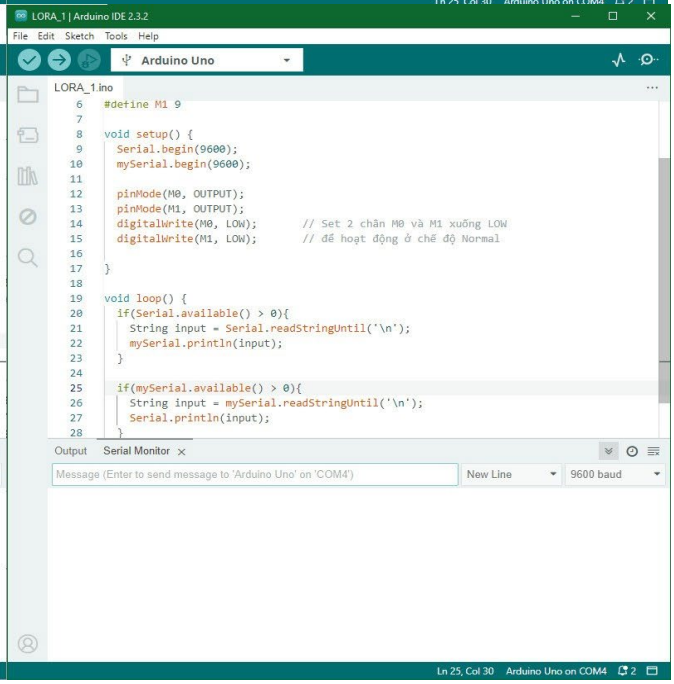
```
  mySerial.begin(9600);

  pinMode(M0, OUTPUT);
  pinMode(M1, OUTPUT);
  digitalWrite(M0, LOW);        // Set 2 chân M0 và M1 xuống LOW
  digitalWrite(M1, LOW);        // để hoạt động ở chế độ Normal

}

void loop() {
  if(Serial.available() > 0){
    String input = Serial.readStringUntil('\n');
    Serial.println(input);
  }
  if(mySerial.available() > 0){
    String input = mySerial.readStringUntil('\n');
    Serial.println(input);
  }
  delay(20);
}
```

Code xong chạy rồi mở cổng Serial sau đó nhập chuỗi vào test chuyền nhận, giống kiểu nhắn tin


LORA_2 | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
LORA_2.ino
mySerial.begin(9600);
pinMode(M0, OUTPUT);
pinMode(M1, OUTPUT);
digitalWrite(M0, LOW); // Set 2 chân M0 và M1 xuống LOW
digitalWrite(M1, LOW); // để hoạt động ở chế độ Normal
void loop() {
if(Serial.available() > 0){
String input = Serial.readStringUntil('\n');
mySerial.println(input);
}
if(mySerial.available() > 0){
String input = mySerial.readStringUntil('\n');
Serial.println(input);
}
delay(20);
Output Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM5')
New Line
9600 baud
Ln 28, Col 4 Arduino Uno on COM5


LORA_1 | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
LORA_1.ino
#define M1 9
void setup() {
Serial.begin(9600);
mySerial.begin(9600);
pinMode(M0, OUTPUT);
pinMode(M1, OUTPUT);
digitalWrite(M0, LOW); // Set 2 chân M0 và M1 xuống LOW
digitalWrite(M1, LOW); // để hoạt động ở chế độ Normal
void loop() {
if(Serial.available() > 0){
String input = Serial.readStringUntil('\n');
mySerial.println(input);
}
if(mySerial.available() > 0){
String input = mySerial.readStringUntil('\n');
Serial.println(input);
}
Output Serial Monitor
hello world
New Line
9600 baud
Ln 25, Col 30 Arduino Uno on COM4


LORA_2 | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
LORA_2.ino
mySerial.begin(9600);
pinMode(M0, OUTPUT);
pinMode(M1, OUTPUT);
digitalWrite(M0, LOW); // Set 2 chân M0 và M1 xuống LOW
digitalWrite(M1, LOW); // để hoạt động ở chế độ Normal
void loop() {
if(Serial.available() > 0){
String input = Serial.readStringUntil('\n');
mySerial.println(input);
}
if(mySerial.available() > 0){
String input = mySerial.readStringUntil('\n');
Serial.println(input);
}
delay(20);
Output Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM5')
New Line
9600 baud
hello world
Ln 25, Col 30 Arduino Uno on COM5


LORA_1 | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
LORA_1.ino
#define M1 9
void setup() {
Serial.begin(9600);
mySerial.begin(9600);
pinMode(M0, OUTPUT);
pinMode(M1, OUTPUT);
digitalWrite(M0, LOW); // Set 2 chân M0 và M1 xuống LOW
digitalWrite(M1, LOW); // để hoạt động ở chế độ Normal
void loop() {
if(Serial.available() > 0){
String input = Serial.readStringUntil('\n');
mySerial.println(input);
}
if(mySerial.available() > 0){
String input = mySerial.readStringUntil('\n');
Serial.println(input);
}
Output Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM4')
New Line
9600 baud
Ln 25, Col 30 Arduino Uno on COM4

Nếu đã chuyền nhận thành công rồi, thì có thể áp dụng cảm biến thôi các bạn có thể xem lại cách đấu nối các bài trước, với chân Digital của DHT11 để giao tiếp arduino thì các bạn có thể tuỳ chọn chân. Code mình đang dùng dưới đây mình dùng chân D5

Code truyền:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(6, 7); //TX, RX
#include <DHT.h>
#define DHTPIN 5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#define M0 8
#define M1 9

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);

  pinMode(M0, OUTPUT);
  pinMode(M1, OUTPUT);
  digitalWrite(M0, LOW);        // Set 2 chân M0 và M1 xuống LOW
  digitalWrite(M1, LOW);        // để hoạt động ở chế độ Normal
  dht.begin();
}

void loop() {
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  mySerial.println("Do Am: " +String(t));
  mySerial.println("Nhiet Do: "+String(h));
  mySerial.println();
  delay(2000);
}
```

Code nhận:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(6, 7); //TX, RX

#define M0 8
#define M1 9

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);

  pinMode(M0, OUTPUT);
  pinMode(M1, OUTPUT);
  digitalWrite(M0, LOW);        // Set 2 chân M0 và M1 xuống LOW
  digitalWrite(M1, LOW);        // để hoạt động ở chế độ Normal

}

void loop() {
  if(mySerial.available() > 0){
    String input = mySerial.readStringUntil('\n');
    Serial.println(input);
  }
}
```

Nạp xong rồi thì các bạn có thể mở Serial Monitor ở bên Nhận để test kết quả. Thậm chí các bạn có thể truyền và nhận cả 2 bên đồng thời.

Nội dung 3: Thực hành các chế độ truyền nhận dữ liệu trong Lora

Dùng Lora ở chế độ 1 (Wake up) đánh thức Lora ở chế độ 2 (Power Saving)

Rất đơn giản bạn chỉ cần thay đổi trạng thái M0 và M1 đúng với yêu cầu khai báo chế độ Lora. Có thể test 2 con Lora, một con Lora chế độ 1, một con Lora ở chế độ 2 sau đó nhắn gửi tin.

Code cho Lora chế độ 1:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(6, 7); //TX, RX

#define M0 8
#define M1 9

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);

  pinMode(M0, OUTPUT);
  pinMode(M1, OUTPUT);
  digitalWrite(M0, LOW);       // Set 2 chân M0 LOW và M1 xuống HIGH
  digitalWrite(M1, HIGH);       // để hoạt động ở chế độ Wake-UP

}

void loop() {
  if(Serial.available() > 0){
    String input = Serial.readStringUntil('\n');
    Serial.println(input);
  }
  if(mySerial.available() > 0){
    String input = mySerial.readStringUntil('\n');
    Serial.println(input);
  }
  delay(20);
}
```

Code cho Lora chế độ 2: Code gần giống với Lora 1 chỉ cần sửa ở chỗ Setup() sửa lại chế độ thành :

```
  digitalWrite(M0, HIGH);       // Set 2 chân M1 LOW và M0 xuống HIGH
  digitalWrite(M1, LOW);       // để hoạt động ở chế độ Power-Savin
```