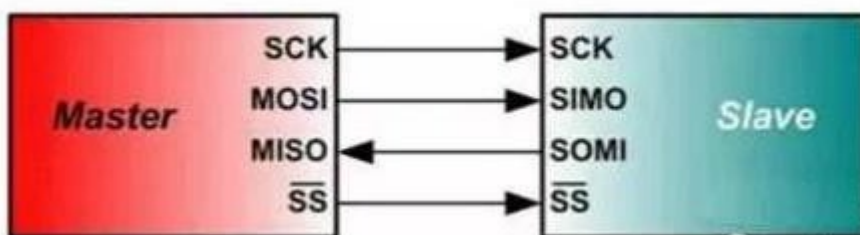


# GIAO THỨC TRUYỀN NHẬN DỮ LIỆU SPI/I2C/UART

## 1. Chuẩn giao tiếp SPI:

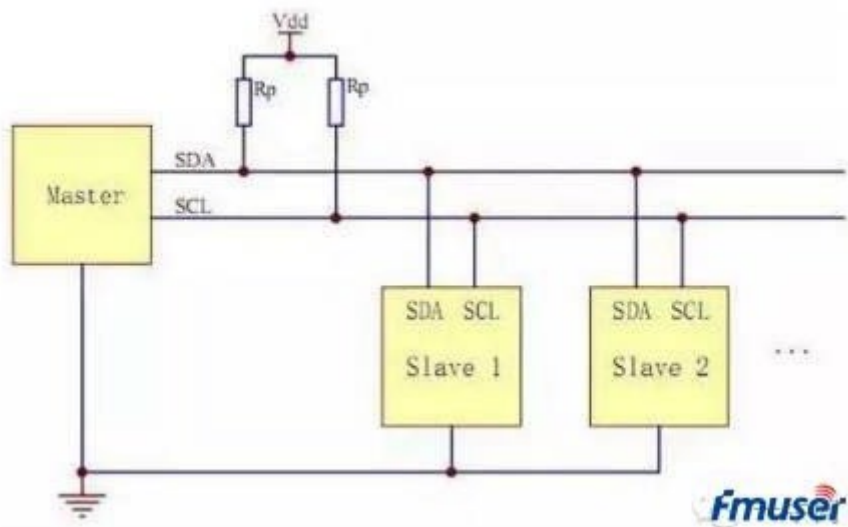
- SPI là viết tắt của Serial Peripheral Interface, một giao thức truyền thông nối tiếp thường được sử dụng trong các hệ thống nhúng để trao đổi dữ liệu tốc độ cao giữa nhiều thiết bị trên bus. Các thiết bị giao tiếp qua SPI có kiến trúc chủ-phụ với nhiều thiết bị phụ được kết nối với một thiết bị chính. Ngoài ra, giao tiếp SPI hỗ trợ giao tiếp song công hoàn toàn, nghĩa là cả chủ và phụ đều có thể truyền và nhận dữ liệu đồng thời



- Không giống như các giao thức truyền thông khác như UART hay I2C, SPI không có giao thức được xác định trước và không có thông số truyền thông cố định. Tính linh hoạt này làm cho SPI trở nên lý tưởng cho các ứng dụng truyền dữ liệu yêu cầu truyền dữ liệu theo thời gian thực hoặc băng thông cao.

## 2. Chuẩn giao tiếp I2C:

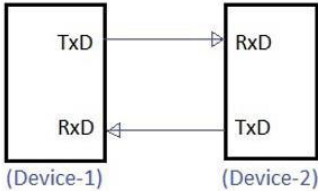
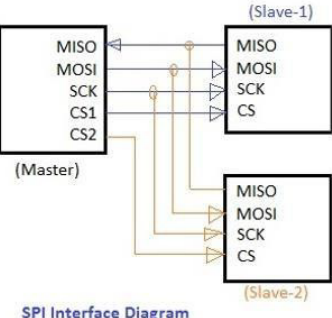
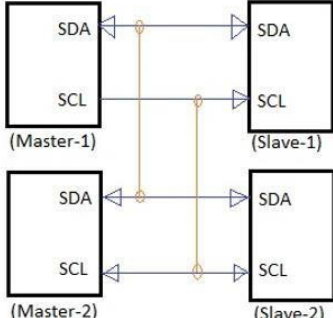
- I2C (viết tắt của Inter-Integrated Circuit), còn được gọi là I2C hoặc IIC, là một bus truyền thông nối tiếp đồng bộ, đa chủ/đa nô lệ. Nó là bus nối tiếp hai dây hai chiều sử dụng dây đồng hồ nối tiếp (SCL) và dây dữ liệu nối tiếp (SDA) để gửi và quản lý dữ liệu từng bit giữa nhiều thiết bị. Với I2C, nhiều nô lệ có thể được kết nối với một chủ duy nhất (như SPI) và nhiều chủ có thể điều khiển một hoặc nhiều nô lệ.
- I2C mang lại khả năng mở rộng và tính linh hoạt cao trong việc kết nối nhiều thiết bị. Tuy nhiên, nó có một vài nhược điểm. Nó hoạt động ở tốc độ chậm hơn so với SPI, đồng hồ và đường dữ liệu cần có điện trở kéo lên. I2C cũng liên quan đến việc xử lý lỗi và logic phức tạp hơn cũng như các vấn đề tiềm ẩn như xung đột bus và nhiễu.



### 3. Chuẩn giao tiếp UART:

- UART hay còn gọi là bộ thu/phát không đồng bộ phổ quát là một trong những giao thức nối tiếp sớm nhất và đơn giản nhất để trao đổi dữ liệu nối tiếp giữa hai thiết bị. Giao thức truyền thông không đồng bộ chỉ sử dụng hai dây, đó là đường truyền (TX) và đường nhận (RX), để truyền và nhận dữ liệu.
- Các thiết bị được kết nối qua UART giao tiếp bằng cách gửi các bit ở tốc độ truyền được xác định trước, thường bao gồm các bit bắt đầu, dừng và các bit chẵn lẻ tùy chọn. Các thiết bị UART không sử dụng tín hiệu đồng hồ chung. Thay vào đó, họ phải thống nhất về tốc độ truyền và định dạng dữ liệu.
- Nhiều nhà phát triển sử dụng UART để kết nối không dây và xử lý máy tính vì tính dễ cài đặt, giao diện thân thiện với người dùng và giá cả phải chăng.

### 4. So sánh giữa các giao thức:

Đặc điểm	UART	SPI	I2C
<u>Sơ đồ kết nối</u>	<p>Kết nối đơn giản</p>  <p>UART Interface Diagram</p>	<p>Kết nối khá phức tạp</p>  <p>SPI Interface Diagram</p>	<p>Kết nối khá đơn giản</p>  <p>I2C Interface Diagram</p>
<u>Số lượng dây</u>	2 dây (1 dây truyền & 1 dây nhận dữ liệu)	4 dây Số lượng dây tăng khi số thiết bị tăng	2 dây
<u>Chế độ truyền</u>	Full duplex (Không phân biệt master – slave)	Full duplex (Một master & nhiều slave)	Half duplex (Nhiều master & nhiều slave)
<u>Tốc độ truyền</u>	Truyền không đồng bộ (Tốc độ tư đặt, tối đa khoảng 460kbps)	Truyền đồng bộ (Tốc độ khoảng 10Mbps đến 20Mbps)	Truyền đồng bộ (Hỗ trợ tốc độ 100kbps, 400kbps, 3.4Mbps, 1Mbps)
<u>Khoảng cách</u>	12m trên lý thuyết	Trên bo mạch	Trên bo mạch
<u>Số thiết bị</u>	2 thiết bị Giao tiếp 1 – 1	Số lượng hạn chế Giao tiếp bằng chân chọn chip	Lên đến 127 thiết bị Giao tiếp bằng địa chỉ

- Chuẩn giao tiếp truyền thông nối tiếp UART trên **Arduino** (hay còn được biết đến với tên gọi Serial) là chuẩn giao tiếp được sử dụng rất nhiều trong các ứng dụng hệ thống nhúng.
- Serial là chuẩn giao tiếp truyền thông nối tiếp không đồng bộ, sử dụng UART (Universal Asynchronous Receiver-Transmitter, một bộ phận phần cứng được tích hợp bên trong chip ATmega328 của board Arduino Uno) kết hợp với mức logic TTL (mức điện áp cao tương ứng với 5V hoặc 3.3V và mức điện áp thấp tương ứng với 0V)
- UART là tên gọi để chỉ một bộ phận phần cứng dùng để chuyển đổi thông tin từ thanh ghi chứa nội dung cần trao đổi ra bên ngoài (thông thường có 8 bits) thành 2 dây tín hiệu Serial, phục vụ cho chuẩn giao tiếp Serial chứ không phải là tên gọi của một chuẩn giao tiếp.

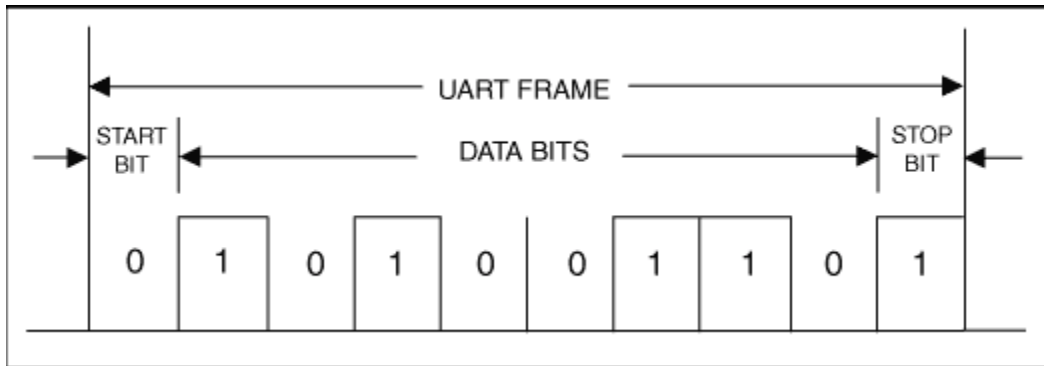
#### 4.1. Baudrate:

- Việc truyền 1 bit data sau mỗi 1 micro giây, ta nhận thấy rằng để việc giao tiếp diễn ra thành công, giữa 2 thiết bị cần có những thông nhất rõ ràng về khoảng thời gian truyền cho mỗi bit dữ liệu, hay nói cách khác tốc độ truyền cần phải được thống nhất với nhau. Tốc độ này được gọi là

tốc độ baud (baudrate), được định nghĩa là số bit truyền trong mỗi giây. Ví dụ, nếu tốc độ baud được sử dụng là 9600, có nghĩa thời gian truyền cho 1 bit là  $1/9600 = 104.167$  micro giây.

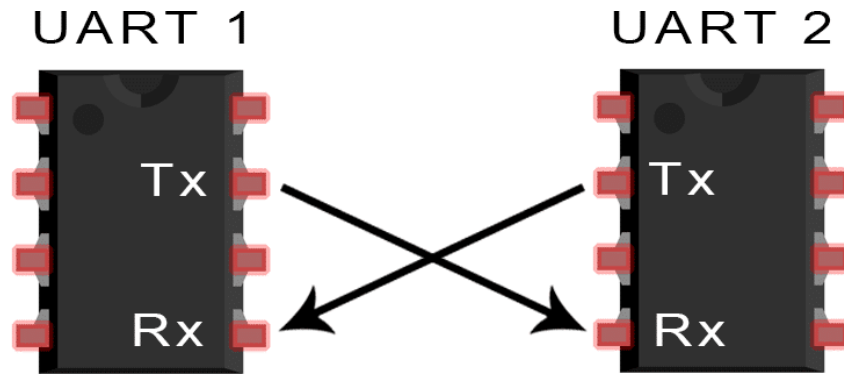
#### 4.2. Khung truyền (frame):

- Bên cạnh tốc độ baud, khung truyền đối với chuẩn giao tiếp Serial cũng hết sức quan trọng để đảm bảo việc truyền nhận được diễn ra chính xác. Khung truyền quy định về số lượng bit truyền trong mỗi lần truyền, bao gồm start bit, các bit data stop bits, ngoài ra còn có thể có parity bit (kiểm tra lỗi dữ liệu trong quá trình truyền nhận).



Hình 1: Ví dụ về khung truyền, bao gồm 1 bit start, 8 bits data và cuối cùng là 1 bit stop.

- Start bit
  - Đây là bit đầu tiên được truyền vào khung truyền, có chức năng thông báo cho thiết bị nhận biết đang có một chuỗi dữ liệu sắp truyền đến. Đối với các dòng AVR nói chung và UnoX nói riêng, ở trạng thái chưa có dữ liệu (Idle) đường truyền luôn kéo lên mức cao. Khi có dữ liệu mới, đường truyền được kéo xuống mức thấp, do đó start bit sẽ được quy định là mức 0.
- Dữ liệu (data)
  - Dữ liệu cần truyền thông thường gồm 8 bits, tuy nhiên chúng ta hoàn toàn có thể tùy chỉnh số lượng bit data cho một gói tin, có thể là 5, 6, 7, 9... Trong quá trình truyền, bit có trọng số thấp nhất (LSB) sẽ được truyền trước, và cuối cùng sẽ là bit có trọng số cao nhất (MSB).
- Stop bits
  - Stop bits là một hoặc nhiều bit có chức năng thông báo cho thiết bị nhận biết rằng một gói dữ liệu đã được gửi xong. Đây là bit quan trọng, cần phải có trong một khung truyền. Giá trị của các stop bit luôn bằng mức Idle (mức nghỉ) và ngược với giá trị của start bit. Đối với các dòng vi điều khiển AVR các bit kết thúc này luôn là mức cao.
- Sử dụng chuẩn giao tiếp Serial với board Arduino Uno
  - Giao tiếp Serial giữa 2 thiết bị với nhau thông thường sẽ có 2 dây tín hiệu, đó là TX (Transmitter) có chức năng truyền tải dữ liệu và RX (Receiver) có chức năng thu nhận dữ liệu. 2 dây tín hiệu này được kết nối chéo nhau giữa 2 thiết bị. Cụ thể, TX của thiết bị 1 kết nối với RX của thiết bị 2 và RX của thiết bị 1 kết nối với TX của thiết bị 2. Điều đặc biệt của Serial đó là quá trình truyền và quá trình nhận dữ liệu được diễn ra hoàn toàn độc lập với nhau.



- Board Arduino đều được hỗ trợ sẵn 1 hoặc nhiều cổng giao tiếp Serial. Đối với board Arduino Uno, cổng Serial được ra chân tại vị trí chân số 1 và chân số 0 trên board), đây được gọi là Hardware Serial, tức là giao tiếp Serial dựa trên phần cứng (có sử dụng UART). Ngoài ra, đối với các vi điều khiển không hỗ trợ UART hoặc trong trường hợp muốn mở rộng nhiều cổng Serial để giao tiếp với nhiều thiết bị có hỗ trợ giao tiếp Serial, ta hoàn toàn có thể giả lập giao tiếp Serial bằng thư viện cho Arduino có tên là SoftwareSerial,

## 5. Thực hành cho vi điều khiển Arduino truyền dữ liệu nhiệt độ độ ẩm lên máy tính thông qua giao thức SPI/I2C

- Việc máy tính lắng nghe dữ liệu từ board Arduino Uno thông thường được sử dụng khi debug chương trình, qua đó chúng ta có thể quan sát và kiểm tra kết quả trả về từ board trong khi hoạt động thông qua cửa sổ Serial Monitor. Một số hàm cần lưu ý khi sử dụng chức năng này:
  - ❖ `Serial.begin(baudrate);`
  - ❖ `Serial.print(message);`
  - ❖ `Serial.println(message);`
- baudrate như đã giới thiệu đó là tốc độ baud, các tốc độ baud thường dùng: 9600, 57600, 115200, 921600,... Lưu ý khi sử dụng: giữa 2 thiết bị giao tiếp với nhau cần được khai báo tốc độ baud như nhau để trao đổi dữ liệu được hiệu quả.
- message ở đây là thông tin mà Arduino muốn truyền qua Serial. Thông tin này có thể là các kiểu dữ liệu như String, int, byte, char, double...
- `Serial.begin(baudrate)` có chức năng khai báo sử dụng Serial với tốc độ baud là baudrate.
- `Serial.print(message)` có chức năng truyền Serial với nội dung là message mà không có kí tự kết thúc dòng. Ta sử dụng hàm này khi cần ghi dữ liệu lên Serial Monitor trên cùng 1 dòng.
- `Serial.println(message)` có chức năng truyền Serial với nội dung là message có kí tự kết thúc dòng. Ta sử dụng hàm này khi cần ghi kết thúc dòng nội dung ghi trên Serial Monitor

### 5.1. Chuẩn bị phần cứng:

- Cảm biến nhiệt độ, độ ẩm DHT11 (hoặc cảm biến khí gas MQ2)
- Board Arduino Uno
- Dây cáp nối arduino và máy tính

### 5.2. Sơ đồ đấu nối:

- Dây cáp nối arduino và máy tính
- Arduino và DHT11: (xem lại Nội dung 1 Buổi 2)

Arduino	DHT11
5V	VCC
GND	GND
D7	Signal (Data)

- Arduino và MQ02:

Arduino	MQ02
5V	VCC
GND	GND
A0	A0

### 5.3. Code tham khảo (DHT):

```
#include "DHT.h" //khai báo thư viện DHT

const int DHTPIN = 7; //khai báo chân dữ liệu DHT
const int DHTTYPE = DHT22; //khai báo kiểu DHT, có 3 loại DHT11, DHT21, DHT22 tùy kết
quả có thể thay loại

DHT dht(DHTPIN, DHTTYPE);

void setup() {
    dht.begin(); // Khởi động cảm biến.
    Serial.begin(9600);
}

void loop() {
    float h = dht.readHumidity(); //Đọc độ ẩm
    float t = dht.readTemperature(); //Đọc nhiệt độ
    delay(1000); //Đợi 1 giây
    Serial.print("Nhiệt độ: ");
    Serial.println(t);
    Serial.print("Độ ẩm: ");
    Serial.println(h);
}
```

### 5.4. Kết quả:

```
Nhiệt độ: 28.40
Độ ẩm: 54.30
Nhiệt độ: 28.40
Độ ẩm: 54.30
Nhiệt độ: 28.40
Độ ẩm: 54.20
Nhiệt độ: 28.40
Độ ẩm: 54.20
```



### 5.5. Code tham khảo (MQ02):

```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(9600);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  int value = analogRead(A0); // Đọc giá trị analog từ chân A0  
  int threshold = 500; // Xét ngưỡng cảnh báo  
  Serial.print("AnalogRead: ");  
  Serial.println(value);  
  if(value < 500) // Nếu vượt quá ngưỡng  
    Serial.println("An toan");  
  else  
    Serial.println("Bao Dong");  
}
```

#### Kết Quả:

```
AnalogRead: 39  
An toan  
AnalogRead: 38  
An toan  
AnalogRead: 38  
An toan  
AnalogRead: 38  
An toan  
AnalogRead: 38  
An toan
```

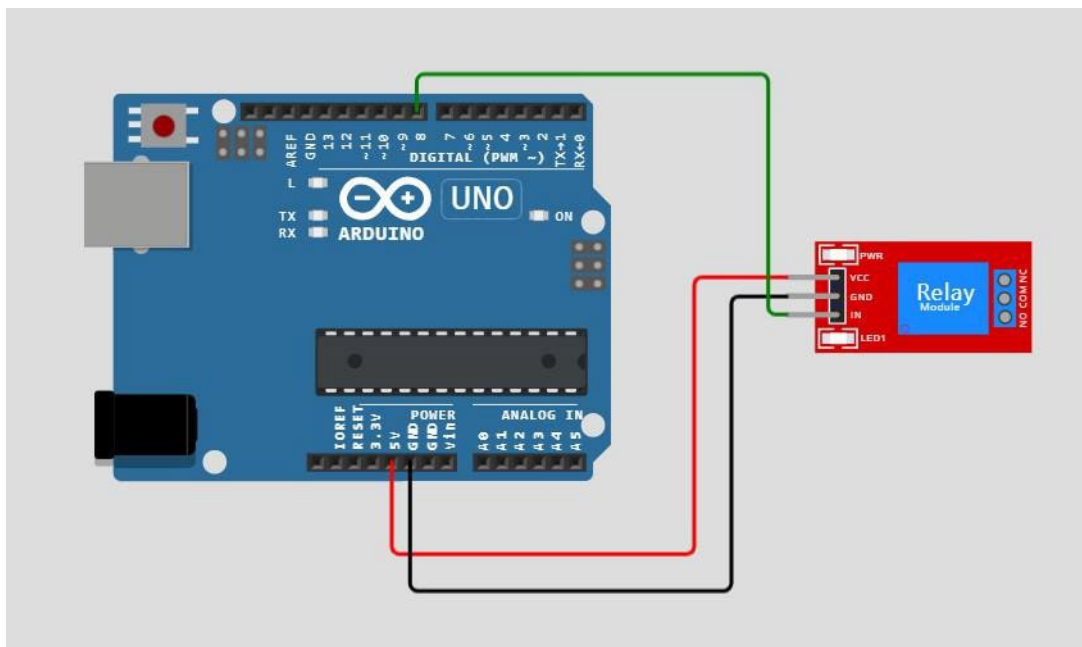
## 6. Thực hành điều khiển bóng đèn bằng giao thức UART giữa máy tính và Arduino

### 6.1. Chuẩn bị phần cứng:

- Bóng đèn + relay
- Board Arduino
- Dây cáp nối arduino và máy tính

### 6.3. Sơ đồ đấu nối

- Dây cáp nối arduino và máy tính
- Arduino-relay-bóng đèn: (xem lại Nội dung 2 Buổi 2)

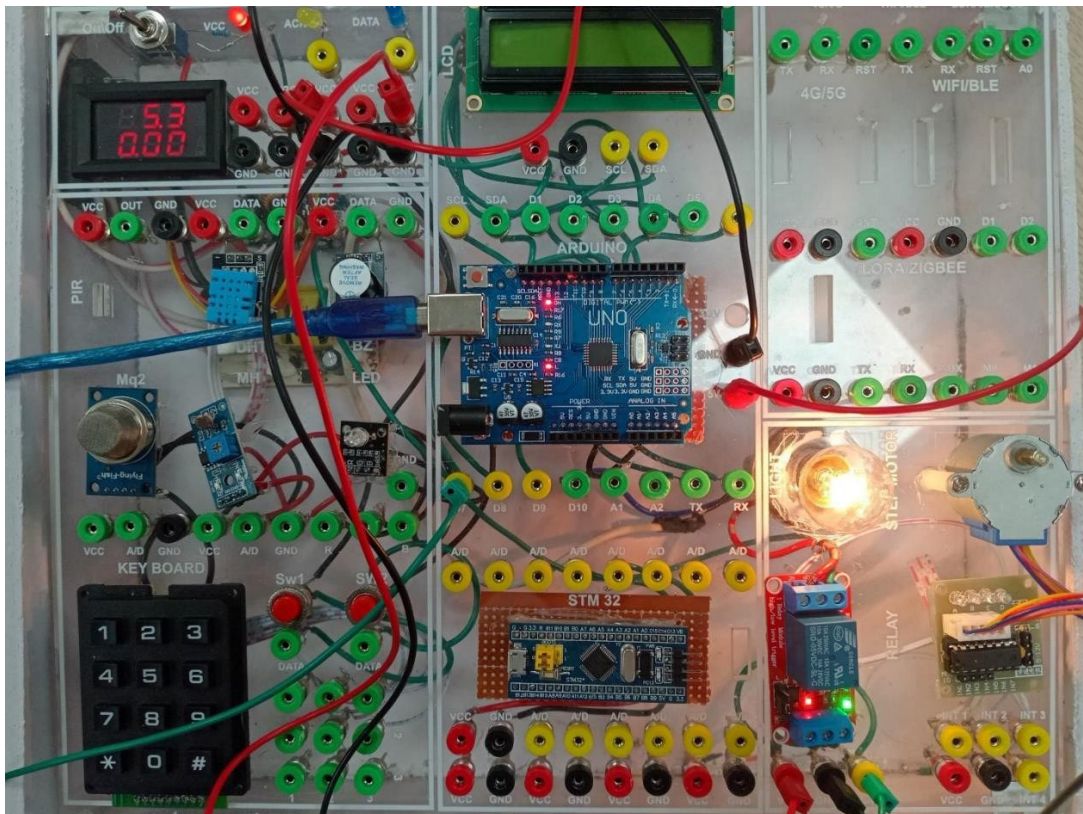


Arduino	Relay
5V	VCC
GND	GND
D8	IN

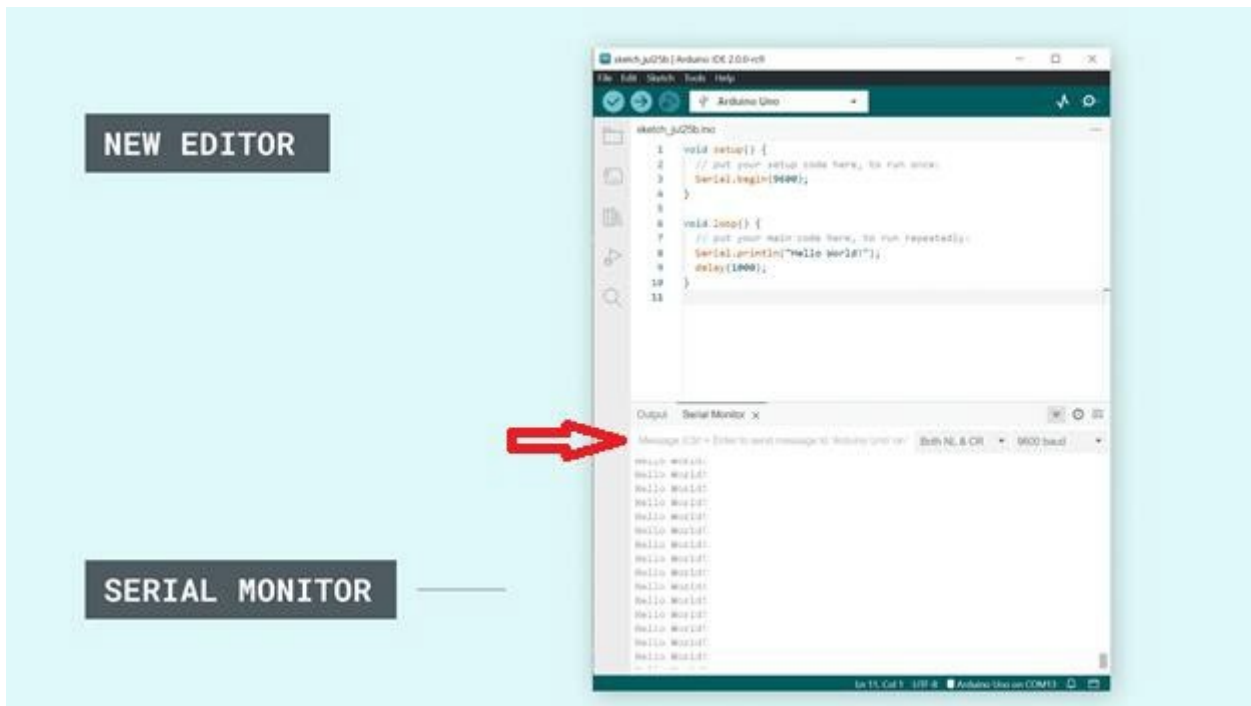
#### 6.4. Code tham khảo:

```
void setup() {  
    Serial.begin(9600); // Khởi tạo Serial với tốc độ Baudrate 9600  
  
    pinMode(8,OUTPUT); //Trên board Uno có một con Led và nó được gắn nối thông qua  
    chân D13  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    if(Serial.available()>0){  
        String str = Serial.readStringUntil('\n'); // Khởi tạo chuỗi để nó lưu từng  
        byte data kí tự, 1 ký tự = 1 byte  
  
        if(str == "on")  
            digitalWrite(8,HIGH); // Nếu chuỗi mình gửi nó nhận là on thì đèn sáng  
        if(str == "off")  
            digitalWrite(8,LOW); // Nếu chuỗi mình gửi nó nhận là off thì đèn tắt  
    }  
}
```

#### 6.5. Kết quả:



- **Lưu ý:** Khi dùng giao thức **UART** từ các chân **Tx, Rx** trước khi nạp code thì hãy ngắt chân **Tx, Rx**. Sau khi nạp xong thì mới đưa vào
- Kết quả: Bật Serial Monitor hoặc Ctrl + Shift + M sau đó gửi thử tin nhắn vào nó trông sẽ như này:
  - Nhập thử **on** hoặc **off** sau đó Enter để xem trạng thái đèn Led có sáng hoặc tắt theo ý mình muốn gửi yêu cầu không



## 7. Thực hành truyền dữ liệu giữa hai Arduino bằng giao thức UART: Thực hành truyền dữ liệu giữa hai Arduino sử dụng giao thức UART, trong đó một Arduino lấy dữ liệu từ cảm biến, truyền tới Arduino thứ hai để hiển thị lên LCD.

- Để hoạt động được, 2 board này cần được cấp nguồn, nguồn này có thể chung hoặc riêng nhưng GND của cả hai board cần được kết nối với nhau.
- Cách truyền nhận dữ liệu Qua Json:
  - Định nghĩa về Json: JSON là một kiểu định dạng dữ liệu trong đó sử dụng văn bản thuần túy, định dạng JSON sử dụng các cặp **key - value** để lưu dữ liệu sử dụng.
  - Chuỗi Json đơn giản gồm 2 phần đó là key và value:
    1. Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}
    2. Một Chuỗi Json gồm 2 phần đó là key và value
      - Key là từ khóa để tìm kiếm và trả về kết quả là value
      - Key (TITLE)
      - Value (Trợ Lý). Value phải đặt trong cặp dấu ngoặc kép "" khi khai báo
    3. Nếu có nhiều dữ liệu (nhiều cặp key => value) thì ta dùng dấu phẩy (,) để ngăn cách
    4. Các key của JSON nên đặt chữ cái không dấu hoặc số, dấu \_ và không có khoảng trắng., ký tự đầu tiên không nên đặt là số

Ví dụ:

```
{"Nhiet_Do":24.5, "Do_Am":63, "Gas": 205}
```

- Thư viện Cần chuẩn bị: "ArduinoJson" tác giả: Benoit Blachon
  - Khai báo thư viện ArduinoJson cho Arduino Ide: #include <ArduinoJson.h>
  - StaticJsonDocument để khai báo Tên và nhóm bộ nhớ chứa tài liệu Json

```
StaticJsonDocument<100> doc;
```
  - doc["ten\_doi\_tuong"] = gia\_tri: Gắn giá trị vào JsonDocument
  - "serializeJson(doc, Serial)": hàm này để in ra tài liệu Json trong Serial Monitor

### 7.1. Chuẩn bị phần cứng:

- DHT + LCD
- Board Arduino 1 + 2
- Dây cáp nối arduino và máy tính

### 7.2. Sơ đồ đấu nối

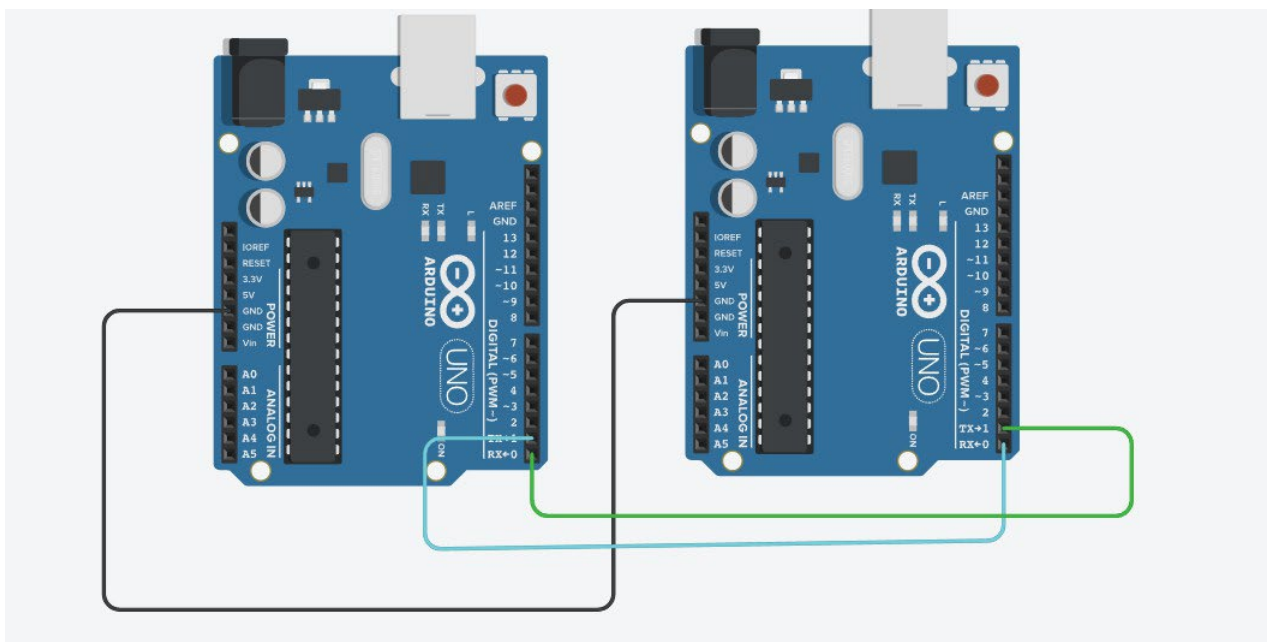
- Arduino 1-DHT: (xem lại Nội dung 1 Buổi 3)

Arduino 1	DHT11
5V	VCC
GND	GND
D8	Signal (Data)

- Arduino 2-LCD: (xem lại Nội dung 1 Buổi 2)

Arduino 2	LCD
5V	VCC
GND	GND
SDA/A4	SDA
SCL/A5	SCL

- 
- Đầu nối 2 arduino



Arduino Uno 1	Arduino Uno 2
Tx	Rx
Rx	Tx
GND	GND

### 7.3. Code Truyền Dữ liệu:

```
#include <ArduinoJson.h>

#include <DHT.h>

#define DHTPIN 8 // Pin Chân Digital Uno để giao tiếp DHT
#define DHTTYPE DHT22 //Loại DHT22 trên IoT Lab dùng

DHT dht(DHTPIN, DHTTYPE);

// Tạo một đối tượng JsonDocument có dung lượng 100 byte
StaticJsonDocument<100> doc;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    dht.begin();
}

void loop() {
    // put your main code here, to run repeatedly:
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();

    // Gán giá trị nhiệt độ và độ ẩm vào đối tượng JsonDocument
    doc["temperature"] = temperature;
    doc["humidity"] = humidity;

    serializeJson(doc, Serial);
    Serial.println();
    delay(1000);
}
```

### 7.4. Code Nhận Dữ liệu:

```
#include <ArduinoJson.h>
#include<LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

StaticJsonDocument<100> doc;
```



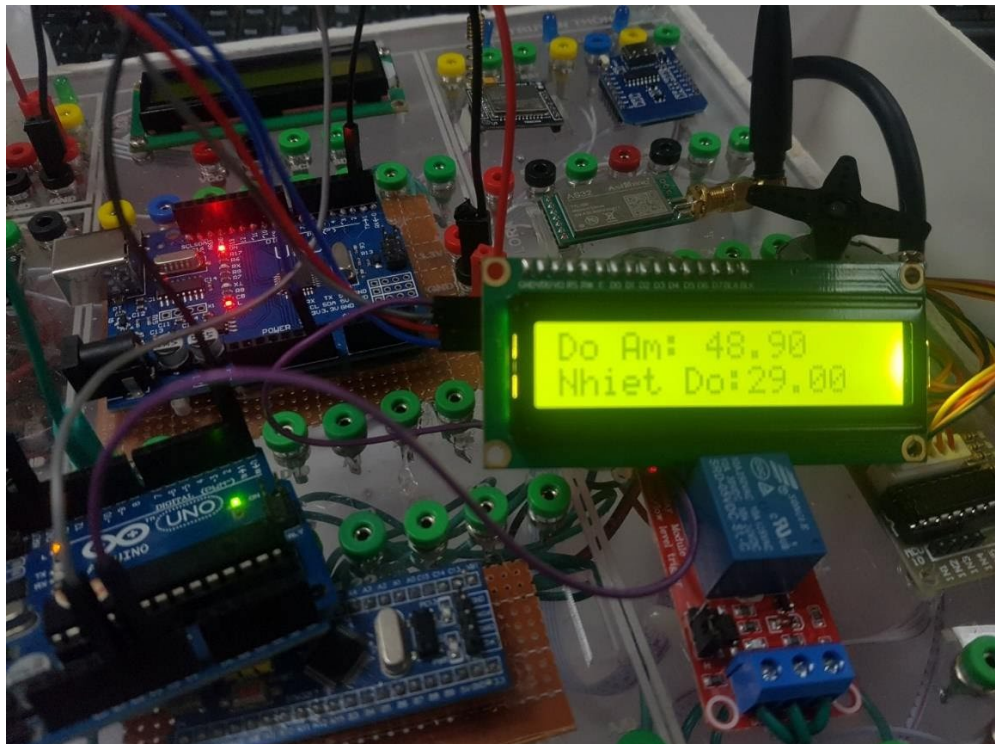
```

void setup() {
    Serial.begin(9600);
    lcd.init();
    lcd.backlight();
}

void loop() {
    while(Serial.available() > 0){
        // Đọc Dữ liệu kênh truyền vừa gửi tới
        String data = Serial.readStringUntil('\n');
        DeserializationError error = deserializeJson(doc, data);
        // Lấy giá trị nhiệt độ và độ ẩm từ đối tượng JsonDocument
        float temperature = doc["temperature"];
        float humidity = doc["humidity"];
        // In các giá trị ra màn hình nối tiếp
        lcd.setCursor(0,0);
        lcd.print("Nhiệt Độ: ");
        lcd.print(temperature);
        lcd.setCursor(0,1);
        lcd.print("Độ Ẩm: ");
        lcd.print(humidity);
        delay(1000);
    }
}

```

## 7.5. Kết quả:





- Lưu ý: Khi nạp code cho 2 con Arduino hãy ngắt Chân Tx(0) và Rx(1) để nạp. Sau khi nạp xong thì cắm lại vào

Kết Quả: Bật “Serial Monitor” hoặc Ctrl + Shift + M

1. Tài liệu Tham khảo:

1. <https://www.facebook.com/photo/?fbid=249915710004369&set=a.143478370648104> (So sánh các giao thức)
2. <https://trolyhocktap.blogspot.com/2020/03/Huong-Dan-Tu-Hoc-Esp8266-Su-dung-Arduino-Json-6.html> (Cách dùng Json cho Arduino)
3. <https://blog.unicloud.com.vn/arduino/arduino-usb-serial/> (Truyền thông nối tiếp UART trên Board Arduino)