**Introduction**

Hello teachers, today our group would like to introduce the project "Remote Control Car with Camera". This is a small, flexible car that can be controlled remotely via a laptop or phone. It combines distance sensors, encoder V2, ESP32-CAM, and the INA219 current sensor.

The project helps our group apply knowledge of programming and electronics, while also developing creativity in optimizing the car's features. We hope that this product will provide an interesting and useful experience for everyone in learning about automation technology.

*Giới thiệu Dự Án "Remote Control Car with Camera"*

Xin chào thầy cô, hôm nay nhóm em xin giới thiệu dự án "Remote Control Car with Camera". Đây là một chiếc xe nhỏ gọn, linh hoạt, có thể điều khiển từ xa qua laptop hoặc điện thoại, kết hợp với cảm biến khoảng cách, encoder V2, ESP32-CAM và cảm biến dòng điện INA219.

Dự án giúp nhóm em áp dụng kiến thức lập trình và điện tử, đồng thời phát triển khả năng sáng tạo trong việc tối ưu hóa tính năng xe. Chúng em hy vọng sản phẩm này sẽ mang lại trải nghiệm thú vị và hữu ích trong việc học hỏi về công nghệ tự động hóa.

**Slide 4 - Hardware (Basic)**

**Động cơ** **3V-9V :** mang lại sự linh hoạt trong việc sử dụng điện áp và rất thích hợp cho các dự án điện tử, mô hình, hoặc các ứng dụng cần điều khiển tốc độ đơn giản.

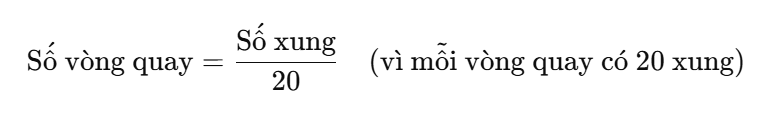
**Dây 18 AWG**: Thích hợp cho các tải có dòng điện từ **2A đến 10A**, phù hợp với các mô-đun, động cơ nhỏ, hoặc các mạch yêu cầu dòng điện vừa phải. Phù hợp với pin 2s2p với thông số 7,4V dung lượng 4000mah - dòng xả 4A.

**Pin Lion :** Điện thế danh định: 3.7V và Điện thế sạc đầy: 4.2V.

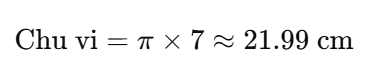
**Slide 5 - Hardware (Controller & Sensor)**

**Encoder** có 20 lỗ trên bánh răng, mỗi lần một lỗ đi qua cảm biến, một xung sẽ được tạo ra. Mỗi xung tương ứng với một bước di chuyển nhỏ của bánh xe.Vì vậy, mỗi vòng quay của bánh xe sẽ tạo ra 20 xung**.**

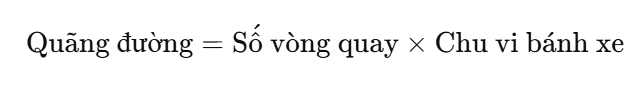
**Số vòng quay này được tính trong 1 khoảng thời gian (Ví dụ 1 giây, 10 giây, vvv)**



**Chu vi của bánh xe**



**Quãng đường tại 1 thời gian nhất định**



**Quãng đường đã di chuyển**

Nhân số vòng quay với chu vi bánh xe, sau đó cộng vào Hàm quãng đường đã di chuyển.

**Vận tốc tại 1 thời gian nhất định**

Quãng đường tại thời gian nhất định / Thời gian nhất định

**L298N**

**L298N là một IC điều khiển động cơ DC và bước (stepper motor) sử dụng cầu H.**

**Cầu H (H-Bridge):** L298N sử dụng mạch cầu H để điều khiển chiều quay của động cơ. Cầu H giúp đảo chiều dòng điện qua cuộn dây động cơ, từ đó thay đổi chiều quay của động cơ.

**Điều khiển bằng tín hiệu logic:** Sử dụng các chân điều khiển (IN1, IN2, IN3, IN4) để điều khiển hướng quay của động cơ.

**Điều chỉnh tốc độ**

* **ENA và ENB** là các chân điều khiển PWM, giúp điều chỉnh tốc độ của động cơ.
* PWM giúp thay đổi **Duty Cycle từ 0 đến 255** để điều khiển tốc độ động cơ.
* Duty Cycle cao (ví dụ: 100%) = động cơ quay nhanh.
* Duty Cycle thấp (ví dụ: 0%) = động cơ tắt

**Điều chỉnh chiều quay**

* **Khi điều khiển động cơ DC**, chân điều khiển (IN1 và IN2) và (IN3 và IN4) sẽ xác định chiều quay của động cơ.
* Điện áp cao (HIGH) và điện áp thấp (LOW) quyết định chiều quay.
* **Động cơ sẽ quay** **theo chiều của dòng điện** (chân HIGH → LOW). Nếu dòng điện chạy từ IN2 → IN1, động cơ trái quay theo chiều tiến.

**Điện áp và dòng điện**

Có thể điều khiển động cơ ở điện áp từ 4.5V đến 36V.

Cung cấp dòng điện lên đến 2A cho mỗi kênh động cơ

**ENA và ENB** là các chân điều khiển PWM, giúp điều chỉnh tốc độ của động cơ.

PWM giúp thay đổi **Duty Cycle từ 0 đến 255** để điều khiển tốc độ động cơ.

Duty Cycle cao (ví dụ: 100%) = động cơ quay nhanh.

Duty Cycle thấp (ví dụ: 0%) = động cơ tắt

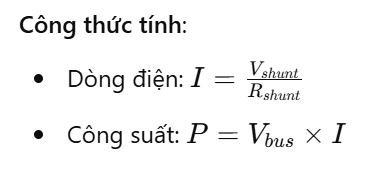
**INA219**

**INA219** là một cảm biến điện áp và dòng điện sử dụng giao thức **I2C** để giao tiếp với vi điều khiển (ví dụ như Arduino hoặc Raspberry Pi). Nó được sử dụng để đo điện áp và dòng điện một cách chính xác và có thể được sử dụng trong các ứng dụng như **theo dõi năng lượng**, **quản lý pin** và các ứng dụng cần đo đạc điện năng.

**Đo dòng điện**: Đo dòng điện một chiều (DC) từ 0 đến 3.2A với độ phân giải 1mA, sử dụng điện trở **shunt 0.1Ω** để đo sự rơi điện áp.

**Đo điện áp**: Đo điện áp từ 0 đến 26V.

**Đo công suất**: Công suất được tính từ công thức P=V×IP = V \times IP=V×I, trong đó VVV là điện áp và III là dòng điện.



**Giao tiếp I2C**: Đơn giản kết nối với vi điều khiển qua giao thức I2C (SDA, SCL).

**Ứng dụng**: Quản lý năng lượng, theo dõi dòng điện, và các hệ thống cần đo đạc điện năng.

Sử dụng thư viện **Adafruit\_INA219** để giao tiếp với cảm biến.

Khi dòng điện chạy qua điện trở shunt, nó tạo ra một điện áp rơi (V\_shunt) tỷ lệ thuận với dòng điện. Cảm biến INA219 đo điện áp này và sử dụng giá trị điện trở shunt để tính toán dòng điện.

**Vbus (Điện áp bus)**: Là điện áp cung cấp cho hệ thống (hoặc điện áp đầu vào của thiết bị). Đơn vị tính của điện áp là **volt (V)**.

**Giao tiếp I2C (Inter-Integrated Circuit) là gì?**

**Giao tiếp I2C (Inter-Integrated Circuit) là** một giao thức truyền thông giữa các vi mạch (IC) sử dụng chỉ 2 dây: SDA (Serial Data) để truyền dữ liệu và SCL (Serial Clock) để điều phối tín hiệu đồng hồ. Một vi điều khiển (Master) có thể giao tiếp với nhiều thiết bị (Slave)(Màn hình LCD, cảm biến,....) qua hai dây này mà không cần kết nối riêng biệt cho từng thiết bị. I2C phổ biến trong các hệ thống điện tử nhỏ, tiết kiệm chân và dễ mở rộng.

**Trong giao tiếp I2C:**

* SDA (Serial Data Line): Dùng để truyền tải dữ liệu giữa các thiết bị. Đây là dây dữ liệu mà các vi mạch sử dụng để gửi và nhận thông tin.
* SCL (Serial Clock Line): Dùng để đồng bộ hóa tín hiệu giữa các thiết bị. Dây này gửi tín hiệu đồng hồ (clock) giúp đảm bảo các thiết bị nhận và gửi dữ liệu đúng thời gian.

**Tóm lại:**

* SDA: Truyền dữ liệu.
* SCL: Đồng bộ hóa dữ liệu.

**HC-SR04**

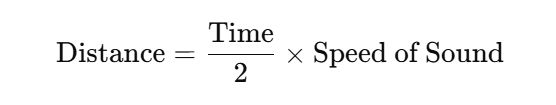
**HC-SR04** là một cảm biến siêu âm dùng để đo khoảng cách. Cảm biến này hoạt động bằng cách phát ra sóng siêu âm và đo thời gian mà sóng này quay lại sau khi gặp vật cản.

### **Thành phần của HC-SR04:**

1. **Truyền siêu âm (Trigger Pin)**: Đây là chân dùng để gửi tín hiệu kích hoạt. Khi nhận được tín hiệu HIGH, nó sẽ phát ra sóng siêu âm.
2. **Nhận siêu âm (Echo Pin)**: Chân này nhận sóng siêu âm phản hồi từ vật cản và tạo tín hiệu HIGH cho vi điều khiển để tính toán khoảng cách.
3. **VCC và GND**: Cung cấp nguồn cho cảm biến.

### **Nguyên lý hoạt động:**

1. **Phát sóng**: Khi bạn gửi tín hiệu HIGH vào chân Trigger, cảm biến sẽ phát sóng siêu âm (tần số 40kHz).
2. **Nhận sóng phản hồi**: Sóng siêu âm sẽ phản xạ lại từ vật cản và cảm biến nhận sóng phản hồi qua chân Echo.
3. **Tính khoảng cách**: Dựa trên thời gian từ lúc phát sóng đến khi nhận sóng (thời gian trễ), cảm biến sẽ tính toán khoảng cách đến vật cản theo công thức:  
   Trong đó:



* + **Time** là thời gian từ khi phát sóng đến khi nhận lại.
  + **Speed of Sound** là vận tốc của sóng siêu âm trong không khí (khoảng 343 m/s ở nhiệt độ phòng).

### **Cách sử dụng:**

* Bạn chỉ cần kích hoạt chân Trigger trong một khoảng thời gian ngắn (10 microseconds), và sau đó đọc thời gian từ chân Echo để tính toán khoảng cách.

### **Kết luận:**

HC-SR04 rất dễ sử dụng và có thể ứng dụng trong các dự án như robot tự hành, đo khoảng cách, và các hệ thống điều khiển tự động.

**III. Model assembly (Lắp ráp mô hình)**

**WebSocket là một giao thức truyền tải dữ liệu hai chiều, hoạt động trên nền tảng TCP. Nó cho phép một kết nối duy nhất giữa client và server mà không cần phải thiết lập lại kết nối mỗi khi có dữ liệu mới. Dưới đây là những điểm chính:**

### **Cách hoạt động:**

1. **Mở kết nối: Client (trình duyệt hoặc ứng dụng) gửi một yêu cầu HTTP đến server để yêu cầu nâng cấp kết nối lên WebSocket.**
2. **Nâng cấp kết nối: Server xác nhận và trả lời một tín hiệu "101 Switching Protocols" để thay đổi giao thức từ HTTP sang WebSocket.**
3. **Giao tiếp liên tục: Sau khi kết nối thành công, client và server có thể gửi và nhận dữ liệu qua lại bất kỳ lúc nào mà không phải thiết lập lại kết nối.**
4. **Đóng kết nối: Khi không còn cần thiết, kết nối có thể được đóng bởi cả client hoặc server.**

### **Đặc điểm của WebSocket:**

* **Giao tiếp hai chiều: Cả client và server đều có thể gửi dữ liệu cho nhau bất cứ lúc nào.**
* **Kết nối liên tục: Sau khi thiết lập kết nối, nó sẽ duy trì mở và không phải tạo lại mỗi lần giao tiếp.**
* **Hiệu suất cao: Giảm bớt độ trễ và tải trên hệ thống vì không phải thực hiện nhiều lần bắt đầu kết nối mới.**
* **Dữ liệu theo thời gian thực: Thường dùng trong các ứng dụng cần cập nhật liên tục như chat, thông báo, game online, hoặc ứng dụng theo dõi dữ liệu trực tuyến (ví dụ: cổ phiếu, tỉ số thể thao).**

### **Ưu điểm:**

* **Tiết kiệm băng thông: Vì không phải gửi yêu cầu HTTP mỗi lần truyền tải dữ liệu, giúp tiết kiệm băng thông.**
* **Thời gian phản hồi nhanh: Thông tin được gửi ngay lập tức mà không cần chờ đợi hoặc làm lại kết nối.**
* **Ứng dụng linh hoạt: Dễ dàng tích hợp vào các ứng dụng cần giao tiếp liên tục và nhanh chóng, như các ứng dụng chat trực tuyến, cập nhật dữ liệu theo thời gian thực, hoặc các hệ thống thông báo.**

### **Các trường hợp sử dụng WebSocket:**

* **Ứng dụng chat: Giao tiếp tức thời giữa người dùng với nhau.**
* **Game trực tuyến: Dữ liệu được trao đổi liên tục giữa người chơi và server.**
* **Thông báo thời gian thực: Cập nhật nhanh các thông báo như thời gian, tin tức hoặc giá chứng khoán.**
* **Dữ liệu đồng bộ: Ứng dụng yêu cầu cập nhật dữ liệu liên tục và ngay lập tức mà không cần người dùng làm mới trang.**

**Tóm lại, WebSocket là một phương pháp truyền tải dữ liệu hiệu quả, nhanh chóng và tiết kiệm băng thông, đặc biệt phù hợp cho các ứng dụng yêu cầu cập nhật thời gian thực.**

**Dùng tụ điện cho ESP32**

**Thầy của bạn đề xuất sử dụng một tụ hóa 12V 470uF để giải quyết vấn đề cấp nguồn cho ESP32.**

### **Giải thích:**

* **Vấn đề bạn gặp phải: Bạn đang dùng động cơ và điều khiển ESP32 bằng module L298N với nguồn 5V, nhưng ESP32 lại gặp vấn đề khi cấp nguồn trực tiếp từ module này (có thể do nguồn không đủ ổn định hoặc không đủ dòng).**
* **Giải pháp: Thầy bạn đề nghị sử dụng một tụ hóa (capacitor) để giảm các biến động điện áp. Cụ thể:**
  + **Tụ hóa 12V 470uF có nhiệm vụ làm mượt dòng điện cung cấp cho ESP32, giúp tránh các biến động điện áp, giúp ESP32 hoạt động ổn định hơn.**
  + **Chân 5V và chân GND của ESP32 cần được kết nối với tụ hóa này, để nó giúp ổn định nguồn điện.**

### **Cách sử dụng tụ hóa:**

* **Chân 5V của ESP32 sẽ nối vào một chân của tụ hóa, và chân GND của ESP32 sẽ nối vào chân còn lại của tụ hóa.**
* **Tụ hóa sẽ giúp hấp thụ và giải phóng điện áp, làm giảm các sóng nhiễu và đảm bảo điện áp ổn định cho ESP32 hoạt động.**

### **Tụ hóa có tác dụng như thế nào?**

* **Công dụng chính: Tụ hóa giúp lọc các biến động điện áp (nhiễu) và giúp duy trì mức điện áp ổn định. Điều này rất quan trọng khi ESP32 cần nguồn điện ổn định để hoạt động, nhất là khi dùng nguồn cấp từ các module như L298N, nơi điện áp có thể không ổn định hoặc có các dao động nhỏ.**
* **Tụ hóa này sẽ giúp giảm thiểu nhiễu và giữ cho điện áp cung cấp cho ESP32 ổn định hơn.Nếu bạn chưa quen với việc nối tụ hóa, bạn có thể thử hình dung: Tụ hóa hoạt động như một "bộ đệm", giúp làm mượt các thay đổi điện áp và đảm bảo rằng ESP32 nhận nguồn điện ổn định để hoạt động hiệu quả.**

**Tụ hóa 12V 470uF được sử dụng để làm mượt điện áp và giúp ổn định nguồn điện. Sau đây là cách thức hoạt động của tụ hóa này:**

1. **Lọc sóng điện áp: Khi có sự thay đổi đột ngột của dòng điện (ví dụ như khi có một thiết bị khác tiêu thụ năng lượng đột ngột), tụ hóa sẽ lưu trữ năng lượng tạm thời và sau đó cung cấp năng lượng trở lại cho mạch khi cần, giúp giảm sự thay đổi của điện áp.**
2. **Giảm nhiễu điện: Tụ hóa còn có tác dụng giảm các nhiễu điện (electrical noise) trong nguồn điện, giúp điện áp cung cấp cho ESP32 được ổn định hơn, làm cho mạch không bị ảnh hưởng bởi các xung nhiễu trong nguồn điện.**
3. **Khả năng chịu điện áp: Tụ hóa 12V 470uF có thể chịu được điện áp lên đến 12V, nhưng vì bạn đang sử dụng nguồn 5V, tụ hóa này sẽ hoạt động an toàn mà không gặp vấn đề về điện áp. Tụ này sẽ ổn định nguồn điện 5V trong mạch của bạn mà không lo bị quá tải hay gây hư hại.**

**Hy vọng điều này giúp bạn hiểu rõ hơn về cách sử dụng tụ hóa để giải quyết vấn đề của bạn!**

### **Mối quan hệ giữa ESP32 và Web thông qua WebSocket:**

#### **1. WebSocket - Khái niệm cơ bản:**

* **WebSocket là một giao thức truyền thông hai chiều cho phép giao tiếp liên tục và thời gian thực giữa client (trình duyệt web) và server (ESP32 trong trường hợp này).**
* **Khi kết nối WebSocket được thiết lập, cả client và server có thể gửi và nhận dữ liệu bất kỳ lúc nào mà không cần phải thiết lập lại kết nối.**

#### **2. WebSocket Client trên Web (HTML, CSS, JavaScript):**

* **Trình duyệt Web sẽ là client trong mô hình này, sử dụng JavaScript để kết nối với ESP32 WebSocket server.**
* **HTML/CSS sẽ đảm nhận giao diện người dùng (UI), trong khi JavaScript sẽ quản lý kết nối WebSocket và truyền dữ liệu giữa client và server.**

#### **3. WebSocket Server trên ESP32:**

* **ESP32 sẽ là server trong mối quan hệ này, chấp nhận kết nối từ client và thực hiện các lệnh dựa trên dữ liệu nhận được từ Web.**
* **ESP32 sẽ nhận dữ liệu điều khiển từ web (ví dụ: "forward", "backward", "brake") và điều khiển các động cơ hoặc gửi lại phản hồi cho web.**

#### **4. Workflow (Quy trình hoạt động):**

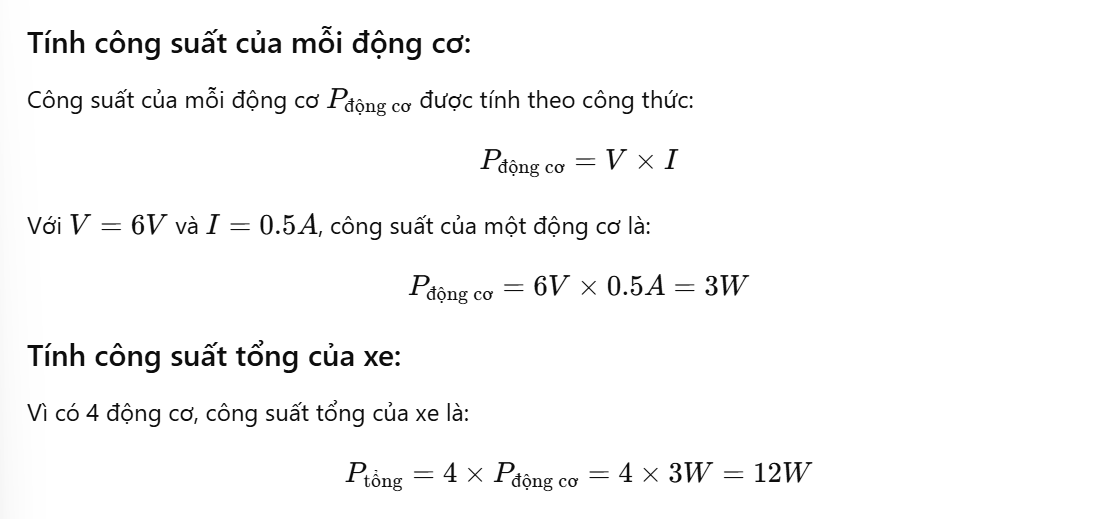
1. **Khởi tạo kết nối WebSocket:**
   * **Web browser (client) sẽ tạo một kết nối đến ESP32 (server) qua WebSocket.**
   * **JavaScript trên web sẽ mở kết nối WebSocket đến ESP32 (chạy trên một địa chỉ IP và cổng nhất định).**
2. **WebSocket kết nối thành công:**
   * **Sau khi kết nối thành công, WebSocket sẽ giữ kết nối mở và dữ liệu có thể được trao đổi trong suốt quá trình.**
3. **Gửi lệnh điều khiển từ Web tới ESP32:**
   * **Khi người dùng nhấn các nút điều khiển trên giao diện Web, JavaScript sẽ gửi thông tin lệnh qua WebSocket đến ESP32 (ví dụ: "move forward", "turn left").**
4. **ESP32 nhận lệnh và thực thi:**
   * **ESP32 sẽ nhận các lệnh từ WebSocket và điều khiển các động cơ tương ứng. Ví dụ: nếu lệnh là "forward", ESP32 sẽ bật động cơ và di chuyển xe về phía trước.**
5. **ESP32 phản hồi lại Web:**
   * **Sau khi thực hiện lệnh, ESP32 có thể gửi phản hồi lại Web (ví dụ: "Moving Forward", "Brake Applied", "Error").**
   * **Phản hồi này có thể được hiển thị trên giao diện Web cho người dùng thấy.**
6. **Đóng kết nối WebSocket:**
   * **Khi người dùng không còn cần điều khiển nữa hoặc khi có sự kiện kết thúc, kết nối WebSocket có thể được đóng lại từ phía Web hoặc ESP32.**

### **1. WebSocket Connection:**

* **Quá trình kết nối:**
  + **Đầu tiên, có một yêu cầu Request từ Client đến Server.**
  + **Sau đó, quá trình Hand Shake (bắt tay) diễn ra giữa client và server để thiết lập kết nối WebSocket. Quá trình này thường chỉ xảy ra một lần khi bắt đầu kết nối.**
  + **Sau khi kết nối WebSocket được thiết lập, kết nối sẽ được duy trì mở và có thể gửi và nhận dữ liệu liên tục giữa client và server mà không cần yêu cầu lại. Điều này giúp giảm độ trễ và tối ưu hiệu suất khi cần trao đổi thông tin thường xuyên hoặc trong thời gian thực.**
* **Đặc điểm:**
  + **Kết nối liên tục: WebSocket giữ kết nối mở cho phép giao tiếp hai chiều mà không cần yêu cầu lại mỗi lần.**
  + **Thích hợp cho các ứng dụng thời gian thực như chat trực tuyến, trò chơi trực tuyến, điều khiển từ xa, v.v.**
  + **Hiệu suất cao: Vì không phải mở lại kết nối cho mỗi yêu cầu mới, WebSocket giúp giảm thiểu độ trễ và tiết kiệm băng thông.**

### **2. HTTP Connection:**

* **Quá trình kết nối:**
  + **Client gửi yêu cầu Request đến Server.**
  + **Server sẽ xử lý yêu cầu và trả về Response cho client.**
  + **Sau khi phản hồi được trả về, kết nối bị đóng. Điều này có nghĩa là mỗi lần client cần gửi yêu cầu mới, một kết nối mới phải được thiết lập và sau đó đóng lại.**
* **Đặc điểm:**
  + **Kết nối không liên tục: Mỗi lần gửi yêu cầu mới, HTTP sẽ mở một kết nối, và sau khi nhận phản hồi từ server, kết nối sẽ bị đóng lại.**
  + **Thích hợp cho các ứng dụng không yêu cầu dữ liệu thời gian thực như tải trang web, gửi biểu mẫu, v.v.**
  + **Độ trễ cao hơn: Do mỗi lần gửi yêu cầu đều phải mở và đóng kết nối, HTTP thường có độ trễ cao hơn so với WebSocket trong các ứng dụng yêu cầu tương tác thời gian thực.**

****

**SDA (Serial Data Line): Chịu trách nhiệm truyền dữ liệu giữa các thiết bị.**

**SCL (Serial Clock Line): Chịu trách nhiệm cung cấp xung đồng hồ để đồng bộ hóa việc truyền nhận dữ liệu.**