**Các Class quan trọng nhất trong thư viện NimBLE:**

**1. NimBLEDevice.h:**

* Class này cung cấp các API để khởi tạo thiết bị BLE, quản lý kết nối và thực hiện các thao tác chung khác.
* **init**
* ...

**2. NimBLEExtAdvertising.h:**

* **NimBLEExtAdvertisement:**
* Thiết lập các thông tin như tên thiết bị, appearance, UUID dịch vụ, cờ (flag), dữ liệu nhà sản xuất, URI, v.v.
* Hỗ trợ cả dữ liệu thô và dữ liệu được cấu trúc.
* **friend class NimBLEExtAdvertising;**
* **NimBLEExtAdvertising : Khởi tạo và hủy quảng cáo, Cấu hình dữ liệu quảng cáo, Kiểm tra trạng thái quảng cáo, Callbacks**
* **Start**
* **setCallbacks** //pCallbacks A pointer to a callback to be invoked when an advertisement stops. //delete callbacks if true callback class will be deleted when advertising is destructed
* **friend class NimBLEDevice;**
* **friend class NimBLEServer;**
* **NimBLEExtAdvertisingCallbacks:**
* **onStopped** //virtual
* **onScanRequest** //virtual

**3. NimBLEServer.h:**

* **NimBLEServer** :Class này cung cấp các API để **Quản lý kết nối, Quản lý dịch vụ, Quảng cáo, Callbacks**
  + NimBLEService\* createService
  + NimBLEExtAdvertising\* getAdvertising()
  + void setCallbacks(NimBLEServerCallbacks\* pCallbacks,bool deleteCallbacks = true); //pCallbacks The callbacks to be invoked
  + friend class NimBLECharacteristic;
  + friend class NimBLEService;
  + friend class NimBLEDevice;
  + friend class NimBLEAdvertising;
* **NimBLEServerCallbacks:** // gồm các virtual callback
  + onConnect(NimBLEServer\* pServer):
  + onDisconnect(NimBLEServer\* pServer
  + onMTUChange(uint16\_t MTU, ble\_gap\_conn\_desc\* desc): Được gọi khi MTU của kết nối thay đổi. MTU là kích thước dữ liệu tối đa có thể truyền trong một gói tin.
  + onPassKeyRequest(): Được gọi khi client yêu cầu passkey để ghép nối. Hàm này trả về passkey mà server sẽ gửi đến client.
  + onAuthenticationComplete(ble\_gap\_conn\_desc\* desc): Được gọi khi quá trình ghép nối hoàn tất. Hàm này có thể được sử dụng để kiểm tra trạng thái mã hóa/ghép nối của kết nối.
  + onConfirmPIN(uint32\_t pin): Được gọi khi sử dụng so sánh mã PIN để ghép nối. Hàm này trả về true để chấp nhận mã PIN do client cung cấp.

**4**. **NimBLEConnInfo** : cung cấp các API trả về địa chỉ của client kết nối với sever

**5. NimBLEService.h:**

* **NimBLEService** :Class này cung cấp các API để Khởi tạo dịch vụ, **Truyền thông với server, Truy cập thông tin dịch vụ, Quản lý đặc tính (characteristic),** 
  + friend class NimBLEServer;
  + friend class NimBLEDevice;

**6. NimBLEUUID.h:**

**7.** **NimBLEAddress.h:**

**8. NimBLECharacteristic.h: //**Đặc tính là đơn vị cơ bản lưu trữ và cung cấp dữ liệu giữa central device (thiết bị trung tâm) và peripheral device (thiết bị ngoại vi) trong giao tiếp BLE.

* **NimBLECharacteristic** :Class này cung cấp các API để Khởi tạo đặc tính với UUID được cung cấp dưới dạng chuỗi ký tự, cùng các tham số tùy chọn về properties (đặc tính) và kích thước dữ liệu tối đa
* Cung cấp các API phục vụ cho việc sử dụng các đặc tính:
  + NIMBLE\_PROPERTY::READ  
    NIMBLE\_PROPERTY::READ\_ENC  
    NIMBLE\_PROPERTY::READ\_AUTHEN  
    NIMBLE\_PROPERTY::READ\_AUTHOR  
    NIMBLE\_PROPERTY::WRITE  
    NIMBLE\_PROPERTY::WRITE\_NR  
    NIMBLE\_PROPERTY::WRITE\_ENC  
    NIMBLE\_PROPERTY::WRITE\_AUTHEN  
    NIMBLE\_PROPERTY::WRITE\_AUTHOR  
    NIMBLE\_PROPERTY::BROADCAST  
    NIMBLE\_PROPERTY::NOTIFY  
    NIMBLE\_PROPERTY::INDICATE

**8. NimBLEClient.h:**

* **NimBLEClient** :Class này cung cấp các API để tương tác với các server trên mạng
  + virtual int connect(IPAddress ip, uint16\_t port) =0;
  + virtual int connect(const char \*host, uint16\_t port) =0;
  + virtual int available() = 0;
  + virtual int read() = 0;
  + virtual int read(uint8\_t \*buf, size\_t size) = 0;
  + virtual int peek() = 0;
  + virtual void flush() = 0;
  + virtual void stop() = 0;
  + virtual uint8\_t connected() = 0;
  + virtual operator bool() = 0;

**9. NimBLEClient.h:**

* **NimBLEScanResults:** để lưu trữ và truy cập danh sách các thiết bị BLE (Bluetooth Low Energy) được tìm thấy trong quá trình quét BLE.
* **NimBLEScan :** thực hiện việc quét các thiết bị BLE (Bluetooth Low Energy) xung quanh. Nó cung cấp các API để bắt đầu, dừng quét, thiết lập các tham số quét và truy cập danh sách các thiết bị được tìm thấy. **Thiết lập callback cho thiết bị.**
* **friend class NimBLEDevice**

**10. NimBLEAdvertisedDevice.h :**

* **NimBLEAdvertisedDevice** đại diện cho một thiết bị BLE được tìm thấy trong quá trình quét BLE. Nó cung cấp các phương thức để truy cập các thông tin chi tiết về thiết bị được quảng cáo, chẳng hạn như địa chỉ BLE, RSSI, dữ liệu nhà sản xuất, tên thiết bị (nếu có), các dịch vụ được hỗ trợ, v.v
* NimBLEAdvertisedDevice();: Khởi tạo một object NimBLEAdvertisedDevice mới (thường được gọi tự động khi tìm thấy thiết bị mới).
* **Truy cập thông tin thiết bị:**
* **Truy cập dữ liệu dịch vụ:**
* **Truy cập các thông tin khác:**
* **NimBLEAdvertisedDeviceCallbacks** :// a scan is performed and \* a new advertised device has been found, we will be called back to be notified
* virtual ~NimBLEAdvertisedDeviceCallbacks()
* virtual void onResult(NimBLEAdvertisedDevice\* advertisedDevice) = 0;
* **NimBLESecurity**: Class này cung cấp các API để bảo mật BLE.

**11. NimBLERemote :**

**Class NimBLERemoteCharacteristic:**

* Đại diện cho một đặc tính GATT từ xa trên một thiết bị BLE được kết nối.
* Cung cấp các API để đọc, ghi, thông báo (notification) và chỉ định (indication) dữ liệu đặc tính.
* Cho phép truy cập các thuộc tính của đặc tính như UUID, giá trị, quyền truy cập, v.v.

**Class NimBLERemoteDescriptor:**

* Đại diện cho một mô tả GATT từ xa được gắn với một đặc tính trên một thiết bị BLE được kết nối.
* Cung cấp các API để đọc và ghi dữ liệu mô tả.

**Class NimBLERemoteService:**

* Đại diện cho một dịch vụ GATT từ xa trên một thiết bị BLE được kết nối.
* Cung cấp các API để truy cập thông tin dịch vụ, tìm kiếm và truy cập các đặc tính GATT thuộc dịch vụ.