**Môn THCNTT3**

**@ Https connection với ESP8266**

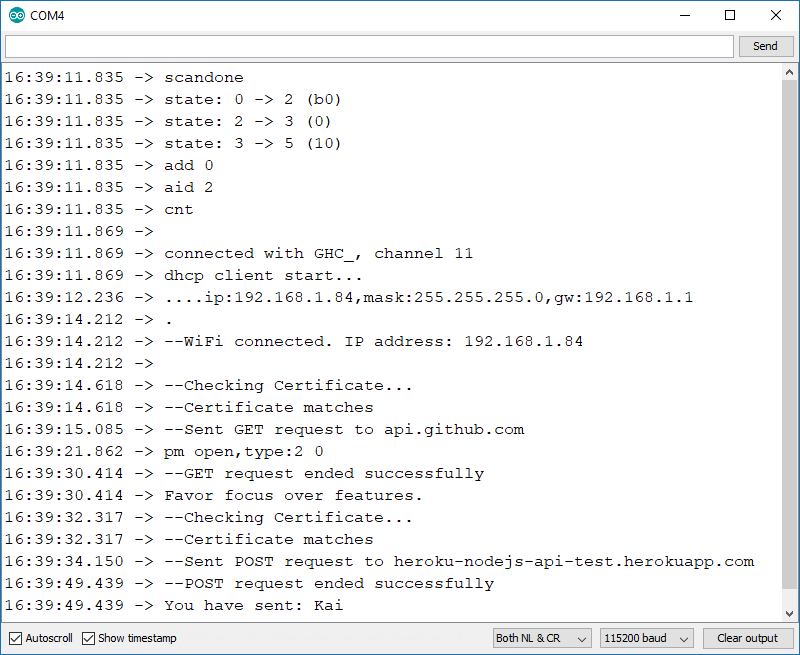
Certificate của https website sẽ được reset sau 90 ngày.

Tôi đã có thể kết nối tới các api dùng giao thức https. Tuy nhiên, tôi gặp vấn đề với việc xác định fingerprint của các https website. Ví dụ, trang api.github.com có thumbprint (lấy từ certificate của chrome) khác hoàn toàn với fingerprint cần để kết nối (lý do có thể liên quan đến việc subdomain có certificate khác domain chính). Tôi quyết định không sử dụng fingerprint để xác thực server cho project này.

Ta có thể đặt Serial.setDebugOutput(true); trong setup() để xem 2 fingerprint khi chúng không đồng nhất thông qua màn hình Serial.

Từ lúc tạo request cho tới lúc nhận response của cả get và post https với esp8266 là hơn **16s**.

Tuy nhiên, chỉ mất **0.5s** để gửi request nếu ta không quan tâm đến response.



String **sendHttpsGet**(WiFiClientSecure httpsClient, String host, int port, String url) {

if (httpsClient.connect(host.c\_str(), port)) {

httpsClient.println("GET " + url + " HTTP/1.1");

httpsClient.println("Host: " + host);

httpsClient.println("User-Agent: ESP8266/Hieu");

httpsClient.println("Connection: close\r\n");

String response = httpsClient.readString();

int bodypos = response.indexOf("\r\n\r\n") + 4;

return response.substring(bodypos);

}

else {

return "ERROR";

}

}

String **sendHttpsPost**(WiFiClientSecure httpsClient, String host, int port, String url, String data) {

if (httpsClient.connect(host, port)) {

httpsClient.println("POST " + url + " HTTP/1.1");

httpsClient.println("Host: " + host);

httpsClient.println("User-Agent: ESP8266/Hieu");

httpsClient.println("Connection: close");

httpsClient.println("Content-Type: **text/plain**; charset=UTF-8");

httpsClient.print("Content-Length: ");

httpsClient.println(data.length());

httpsClient.println();

httpsClient.println(data);

delay(10);

String response = httpsClient.readString();

int bodypos = response.indexOf("\r\n\r\n") + 4;

return response.substring(bodypos);

}

else {

return "ERROR";

}

}

**@ Gửi một chuỗi string giữa 2 Arduino**

Khi dùng serial để giao tiếp giữa 2 Arduino, ta phải đảm bảo chúng sử dụng chung 1 nguồn điện (ví dụ như được lấy từ cùng một máy tính…), nếu không thì Serial không hoạt động.

Vì tốc độ loop của arduino nhanh hơn quá trình truyền Serial, nên khi ta truyền 1 string dài bằng Serial.write() thì khi Arduino còn lại nhận bằng Serial.read(), nó sẽ chỉ đọc được một vài character đầu rồi thực thi tiếp các dòng lệnh tiếp theo. Vì vậy, tôi xử dụng hàm while để chờ đến khi nào Serial đã gửi xong mới cho phép aruidno được loop tiếp.

Fix:

String readSerial() {

String str = "";

while(Serial.available()) {

str += (char) Serial.read();

delay(1);

}

return str;

}

Tuy nhiên, trong quá trình arudino dừng loop để chạy hàm while, nếu các cổng Serial khác được gửi signal sang thì signal đó sẽ bị mất. Arduino không thể đồng thời nhận 2 Serial signals từ 2 cổng khác nhau.

**@ Thư viện ArduinoQueue**

Có nhiều thư viện hỗ trợ queue trong môi trường C++, tôi sẽ dùng nó để enqueue signal, và cứ sau 1 giây sẽ dequeue 1 signal và đồng thời gửi signal đó sang cho nodeMCU. Mục đích là để tránh việc nodeMCU không kịp gửi hết post request chứa signal đến server.