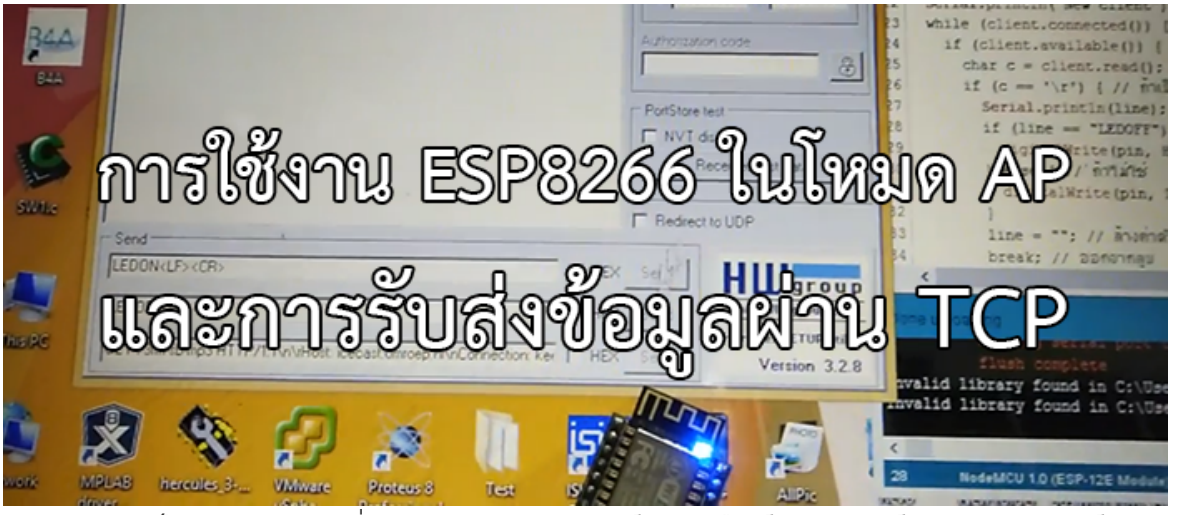
การใช้งาน ESP8266 ในโหมด AP และการรับส่งข้อมูลผ่าน TCP

http://www.ioxhop.com/article/43/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99-esp8266-%E0%B9%83%E0%B8%99%E0%B9%82%E0%B8%AB%E0%B8%A1%E0%B8%94-ap-%E0%B9%81%E0%B8%A5%E0%B8%B0%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%A3%E0%B8%B1%E0%B8%9A%E0%B8%AA%E0%B9%88%E0%B8%87%E0%B8%82%E0%B9%89%E0%B8%AD%E0%B8%A1%E0%B8%B9%E0%B8%A5%E0%B8%9C%E0%B9%88%E0%B8%B2%E0%B8%99-tcp

HTML Content



ESP8266 เป็นบอร์ด WiFi ที่สามารถทำงานได้ 3 โหมด คือ โหมด AP โหมด STA และโหมด AP & STA ในแต่ละโหมดมีความแตกต่างกันดังนี้

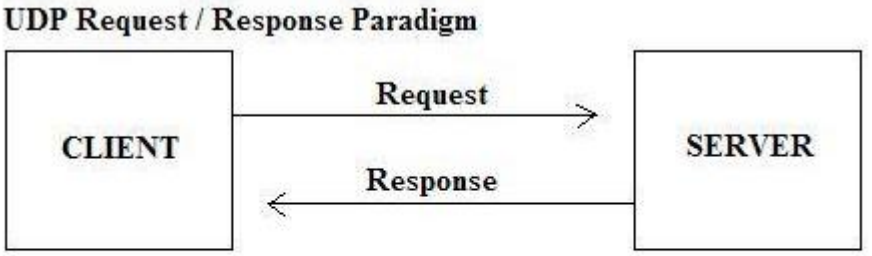
- โหมด AP - เป็นโหมดที่จะต้องรอให้มีอุปกรณ์มาเชื่อมต่อจึงจะสามารถรับส่งข้อมูลกันได้
- โหมด STA - เป็นโหมดที่กำหนดให้ ESP8266 ไปเชื่อมต่อกับอุปกรณ์อื่น ๆ เช่น เร้าเตอร์ แล้วรับส่งข้อมูลระหว่างเครื่องในวงแลนได้
- โหมด AP & STA - เป็นโหมดที่สามารถทำงานได้ทั้ง 2 อย่างภายในเวลาเดียวกัน แต่ความสเถียรจะลดลง และทำให้ใช้กำลังไฟฟ้ามากขึ้น

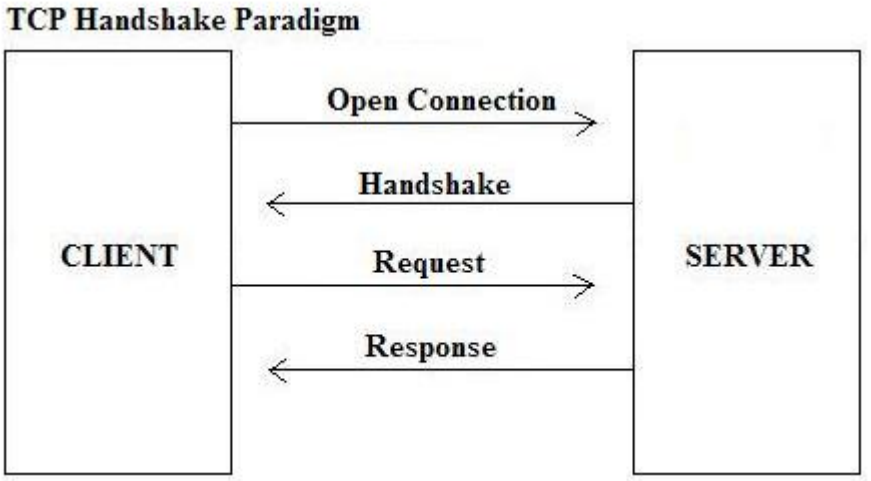
ในการใช้งานควบคุมอุปกรณ์ต่าง ๆ ที่อยู่ในระยะใกล้ และต้องย้ายสถานที่ใช้งานที่บ่อย เช่น นำไปใช้งานควบคุมหุ่นนต์ ควรจะใช้งานในโหมด AP

โปรโตคอลที่ใช้รับ-ส่งข้อมูล

การรับส่งข้อมูลผ่านอินเตอร์เน็ต เช่น การเข้าเว็บไซต์ การใช้บริการเกมส์ออนไลน์ จะใช้โปรโตคอลพื้นฐานอยู่ 2 ตัว คือ TCP และ UDP โดยทั้ง 2 โปรโตคอลมีความแตกต่างกันดังนี้

- TCP - เป็นโปรโตคอลที่ใช้ในการเรียกหน้าเว็บไซต์ โปรโตคอลนี้เมื่อส่งข้อมูลไปแล้วจะต้องรอการยืนยันได้รับข้อมูลจากเครื่องปลายทาง จึงจะเริ่มส่งข้อมูลต่อไป ข้อดีของโปรโตคอลนี้คือได้รับข้อมูลที่ถูกต้องสมบูรณ์
- UDP - เป็นโปรโตคอลที่ใช้สำหรับการรับส่งข้อมูลแบบเรียวทามที่ไม่ต้องการความถูกต้องของข้อมูลมากนัก เช่น การใช้รับส่งข้อมูในเกมส์ออนไลน์ โปรโตคอลนี้จะมีหน้าที่ส่งข้อมูลเพียงอย่างเดียว ปลายทางไม่ต้องยืนยันได้รับข้อมูล ข้อดีของโปรโตคอลนี้คือมีขั้นตอนการทำงานที่ง่ายกว่าแบบ TCP ทำให้ทำงานได้เร็วกว่านั่นเอง





ขอบคุณรูปภาพจาก www.ni.com

ในบทความนี้จะเลือกใช้โปรโตคอล TCP เพราะความถูกต้องของข้อมูลสำคัญมากกว่าความเร็วในงาน IoT และงานควบคุมอุปกรณ์ต่างๆ

คำสั่งที่ควรรู้

WiFiServer::WiFiServer(port);

เป็นคำสั่งเปิดออปเจ็คใหม่เพื่อเริ่มใช้งาน TCP Server ตรงค่าพารามิเตอร์ port สามารถตั้งได้เองโดยต้องเป็นตัวเลขเท่านั้น แนะนำ 10 - 9999 ไม่ควรใช้พอร์ต 80 443 21 22

void WiFiServer::begin(void);

เป็นคำสั่งที่ต้องอยู่ใน void setup() ใช้สำหรับสั่งให้ TCP Server เริ่มการทำงาน

WiFiClient WiFiServer::available(void);

เป็นคำสั่งที่จะมีการให้ค่าของออปเจ็ค WiFiClient **ออกมา เพื่อตรวจสอบว่าขณะนี้มีคนเชื่อมต่อเข้ามาหรือไม่**

bool WiFiClient::connected(void);

ใช้ตรวจสอบว่าขณะนี้ยังเชื่อมต่ออยู่หรือไม่

int WiFiClient::available(void);

ใช้ตรวจสอบว่ามีการส่งข้อมูลเข้ามาแล้วหรือไม่ โดยจะให้ค่ากลับเป็นขนาดข้อมูลที่ถูกเก็บไว้ในบัพเฟอร์

char WiFiClient::read(void);

เป็นคำสั่งอ่านข้อมูลออกมาจากบัพเฟอร์ทีละไบต์

ตัวอย่างโปรแกรมใช้งาน

โค้ดโปรแกรมด้านล่างนี้จะมีการเก็บข้อมูลไว้ที่ตัวแปร line **โดยข้อมูลที่ส่งมาเมื่อจบแล้วจะต้องมีการขึ้นบรรทัดใหม่โดยใช้** \r\n **จากนั้นจะมีการปิดการเชื่อมต่ออัตโนมัติ**

```
#include <ESP8266WiFi.h>



WiFiServer server(88); // ประกาศสร้าง TCP Server ที่พอร์ต 88



String line;



void setup() {

Serial.begin(115200); // เปิดใช้การ Debug ผ่าน Serial

WiFi.mode(WIFI_AP); // ใช้งาน WiFi ในโหมด AP

WiFi.softAP("ESP_IOXhop"); // ตั้งให้ชื่อ WiFi เป็น ESP_IOXhop
```

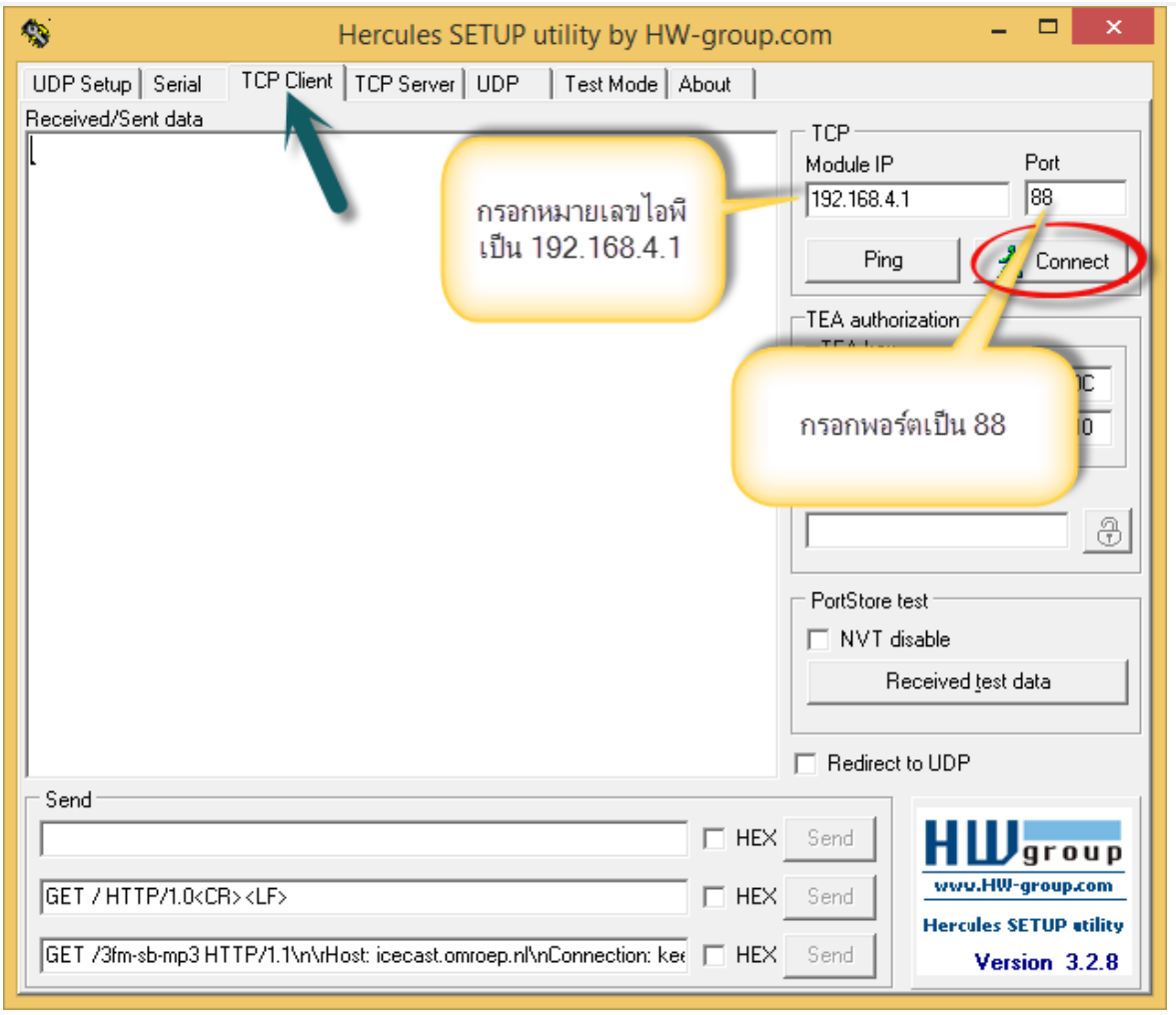
```
server.begin(); // เริ่มต้นใช้ TCP Server
}

void loop() {
WiFiClient client = server.available();
if (!client) // ถ้าไม่มีการเชื่อมต่อมาใหม่
return; // ส่งลับค่าว่าง ทำให้ลูปนี้ถูกยกเลิก

Serial.println("New client"); // ส่งข้อความว่า New client ไปที่ Serial Monitor
while (client.connected()) { // วนรอบไปเรื่อย ๆ หากยังมีการเชื่อมต่ออยู่
if (client.available()) { // ถ้ามีการส่งข้อมูลเข้ามา
char c = client.read(); // อ่านข้อมูลออกมา 1 ไบต์
if (c == '\r') { // ถ้าเป็น \r (return)
Serial.println(line); // แสดงตัวแปร line ไปที่ Serial Monitor
line = ""; // ล้างค่าตัวแปร line
```

```
break; // ออกจากลูป

} else if (c == '\n') { // ถ้าเป็น \n (new line)

// Pass {new line}

} else { // ถ้าไม่ใช่

line += c; // เพิ่มข้อมูล 1 ไบต์ ไปต่อท้ายในตัวแปร line

}

}

}



delay(1);

client.stop(); // ปิดการเชื่อมต่อกับ Client

Serial.println("Client disconnect"); // ส่งข้อความว่า Client disconnect ไปที่ Serial Monitor

}
```

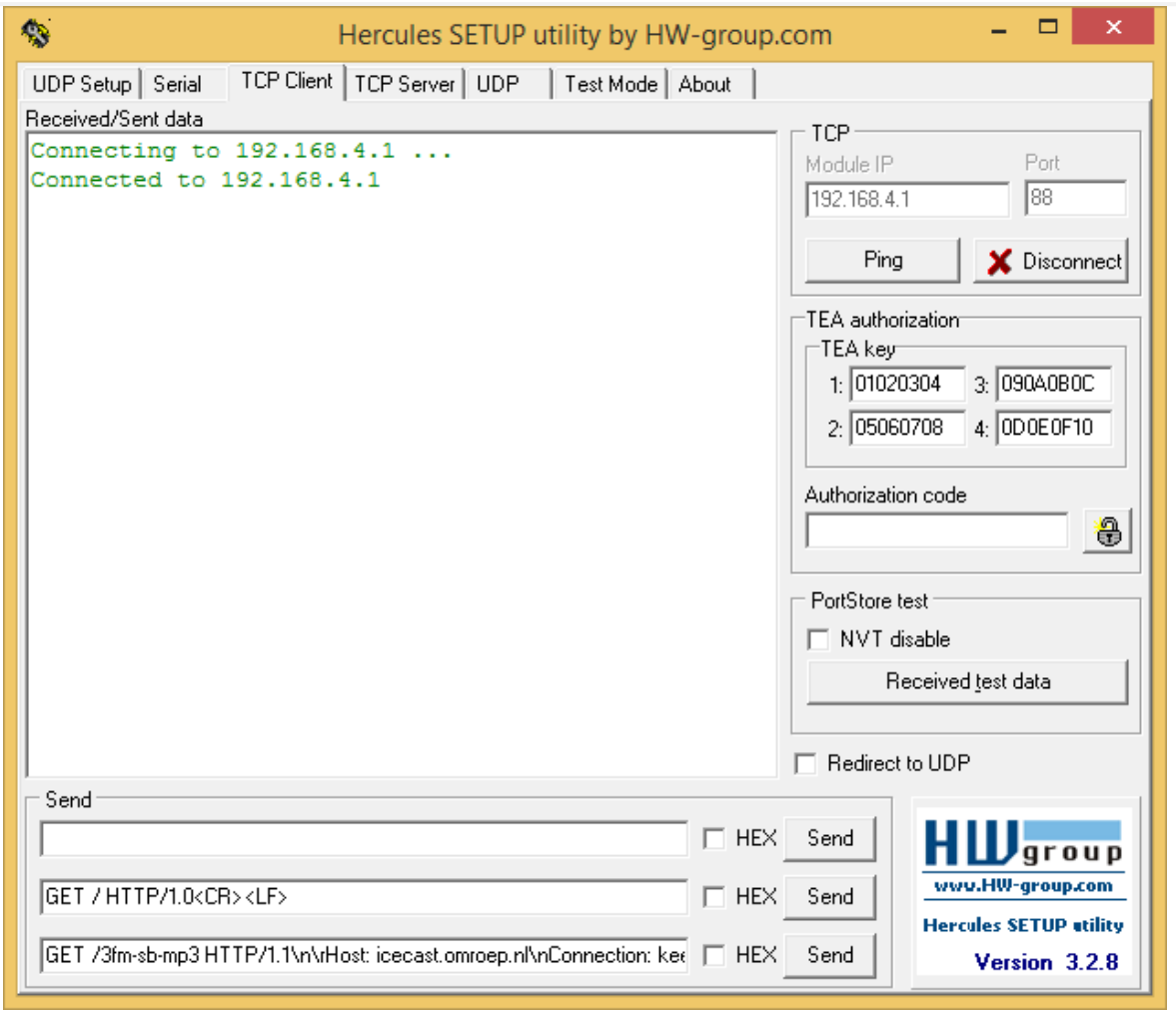
ESP8266_TCPServer.ino hosted with ❤ by GitHub view raw

การทดสอบรับ - ส่งข้อมูล

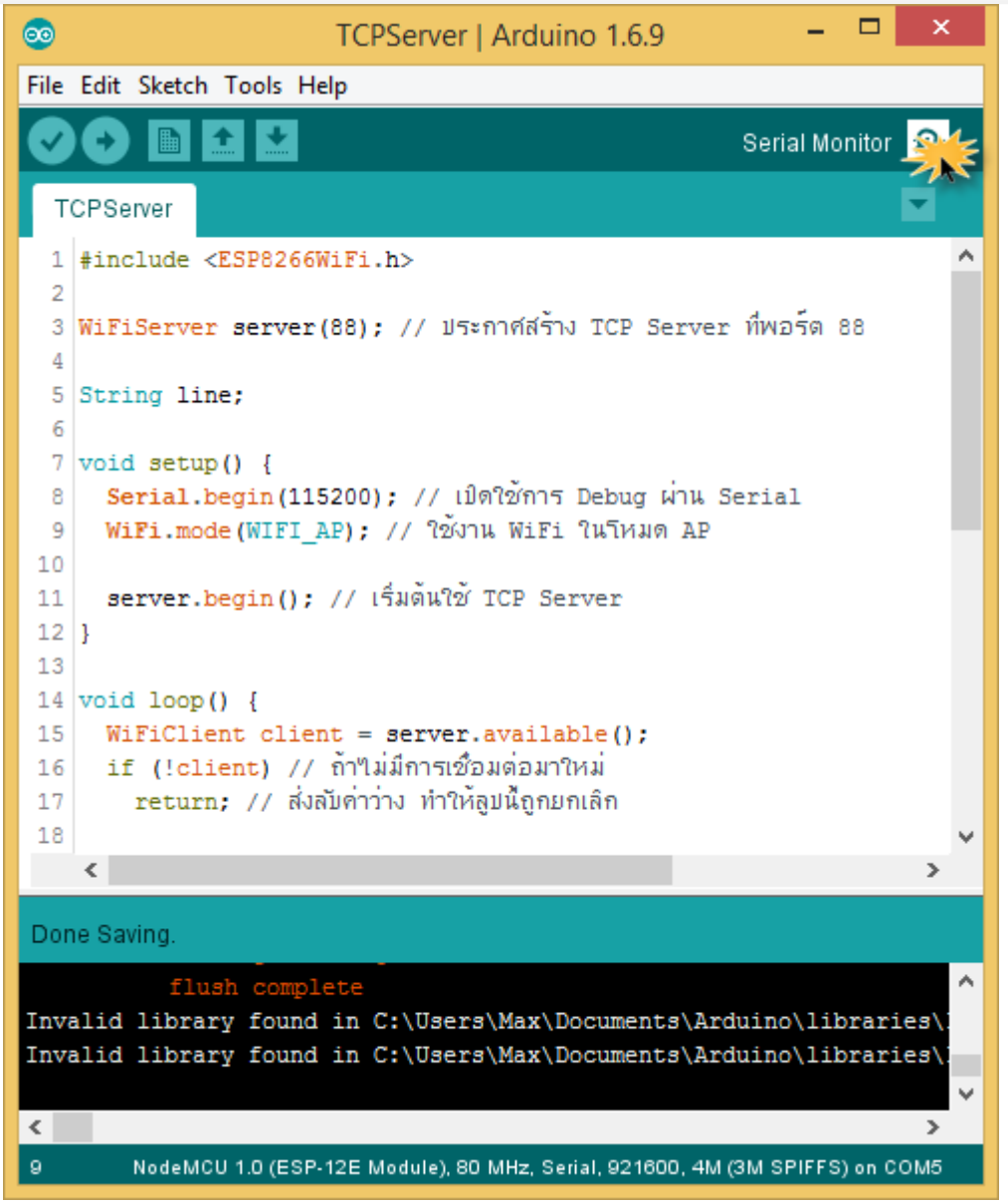
เชื่อมต่อ WiFi เข้าไปที่ตัว ESP8266 ให้เรียบร้อย (ชื่อ WiFi จะชื่อ ESP_xxxx)
ดาว์นโหลดโปรแกรม hercules ได้ที่ hercules_3-2-8.exe เปิดโปรแกรมขึ้นมา แล้วกดไปที่เมนู TCP Client กรอก Module IP เป็น 192.168.4.1 จากนั้นกรอก Port เป็น 88 เสร็จแล้วกด Connect



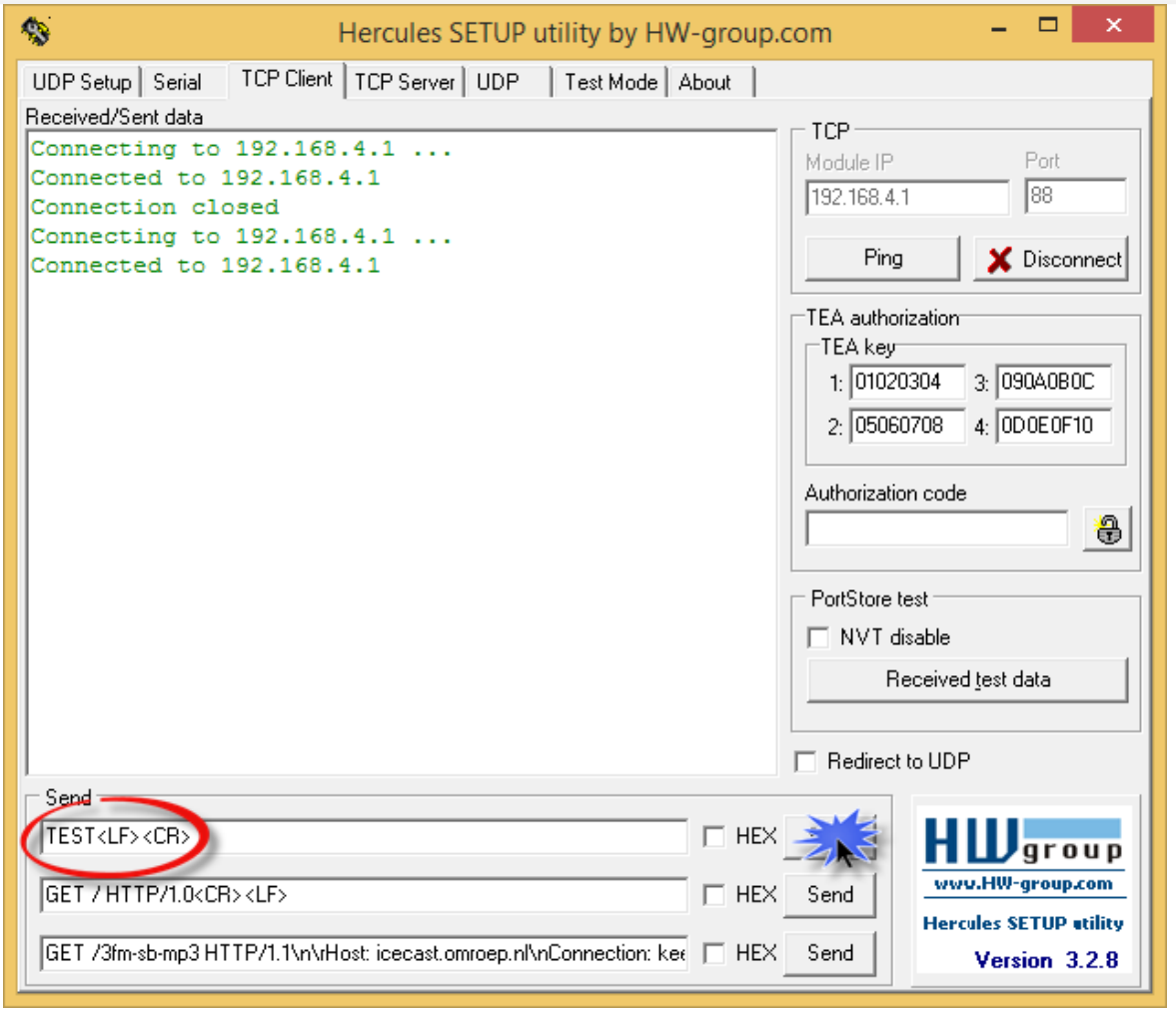
กรณีเชื่อมต่อสำเร็จจะขึ้น Connected **ดังรูป**

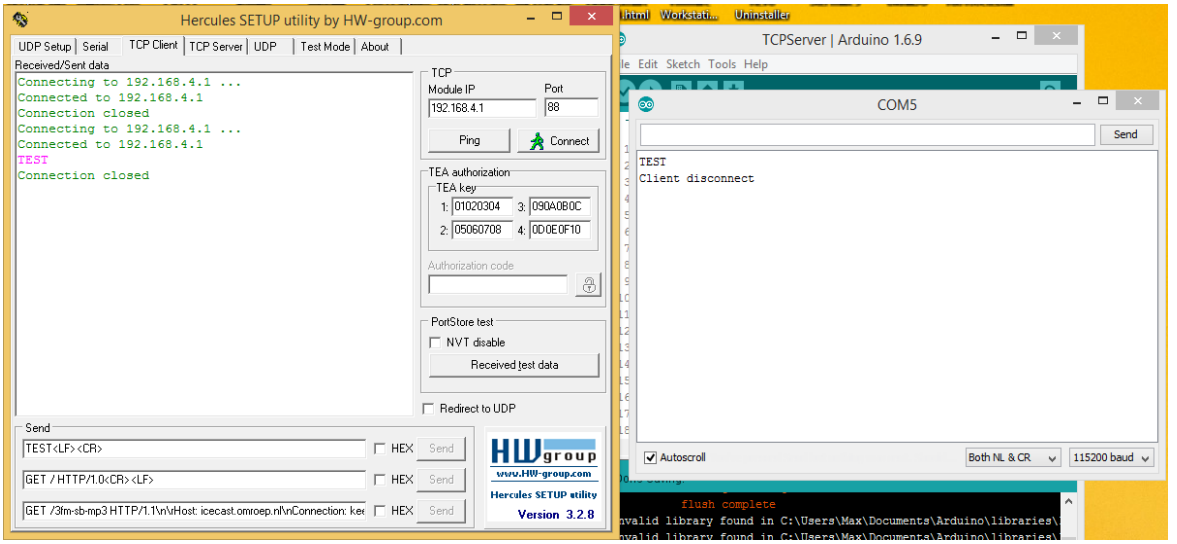


เปิด Serial Monitor ขึ้นมา



ในช่อง Send ลองพิมพ์ TEST<LF><CR> แล้วกด Send จะพบว่าสามารถส่งข้อมูลไปหา ESP8266 ได้แล้ว และมีการตัดการเชื่อมต่ออัตโนมัติ





ทดลองควบคุมหลอด LED

นำโค้ดด้านล่างนี้อัพโหลดเข้าบอร์ด ESP8266

```
#include <ESP8266WiFi.h>
```

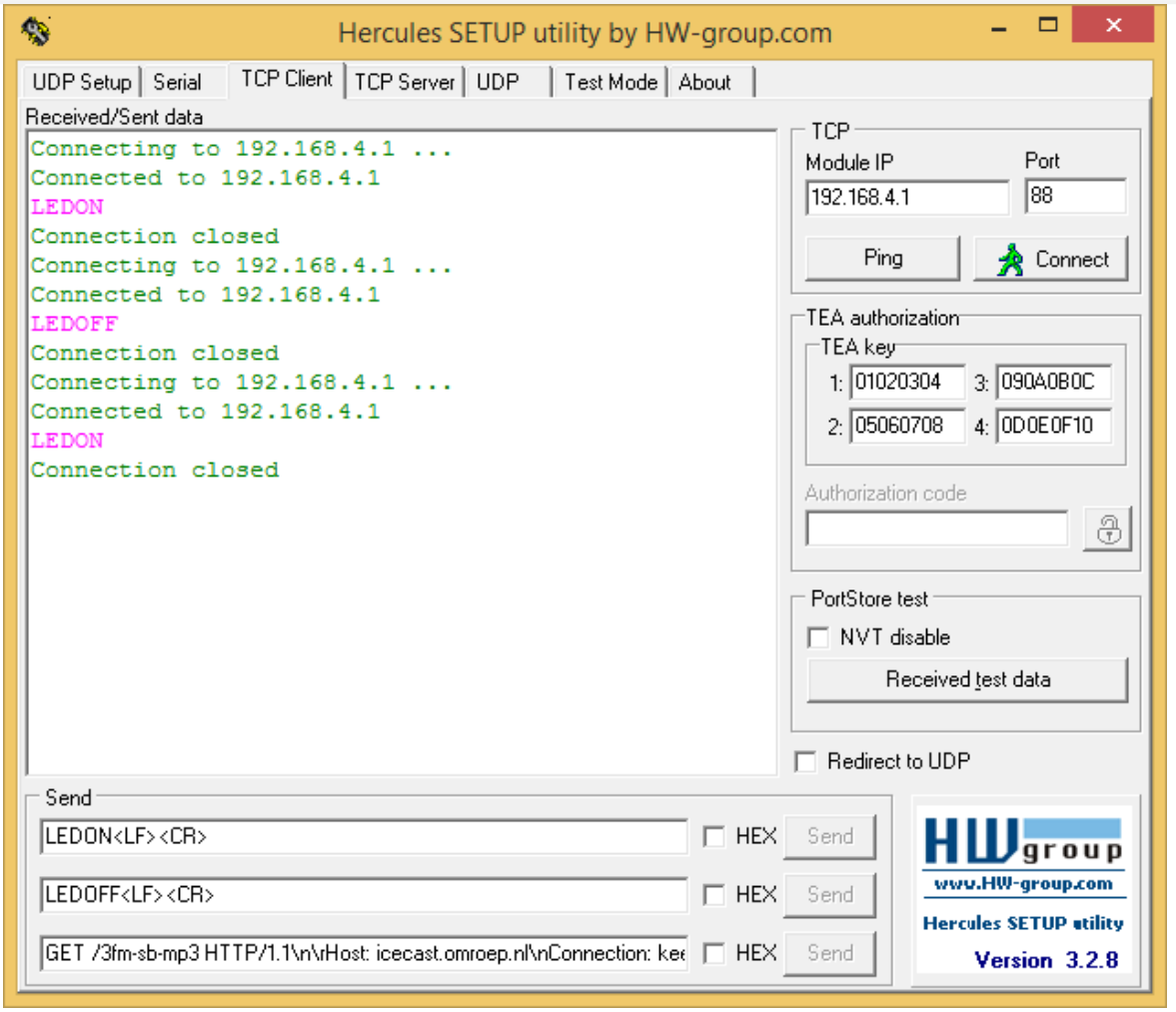
```
WiFiServer server(88); // ประกาศสร้าง TCP Server ที่พอร์ต 88



int pin = 2;

String line;



void setup() {

pinMode(pin, OUTPUT);



Serial.begin(115200); // เปิดใช้การ Debug ผ่าน Serial

WiFi.mode(WIFI_AP); // ใช้งาน WiFi ในโหมด AP

WiFi.softAP("ESP_IOXhop"); // ตั้งให้ชื่อ WiFi เป็น ESP_IOXhop



server.begin(); // เริ่มต้นใช้ TCP Server

}
```

```
void loop() {

WiFiClient client = server.available();

if (!client) // ถ้าไม่มีการเชื่อมต่อมาใหม่

return; // ส่งลับค่าว่าง ทำให้ลูปนี้ถูกยกเลิก



Serial.println("New client"); // ส่งข้อความว่า New client ไปที่ Serial Monitor

while (client.connected()) { // วนรอบไปเรื่อย ๆ หากยังมีการเชื่อมต่ออยู่

if (client.available()) { // ถ้ามีการส่งข้อมูลเข้ามา

char c = client.read(); // อ่านข้อมูลออกมา 1 ไบต์

if (c == '\r') { // ถ้าเป็น \r (return)

Serial.println(line); // แสดงตัวแปร line ไปที่ Serial Monitor

if (line == "LEDON") { // ถ้าสงข้อความเข้ามาว่า LEDON

digitalWrite(pin, HIGH); // ให้ LED ติด

} else { // ถ้าไม่ใช่

digitalWrite(pin, LOW); // ให้ LED ดับ

}
```

```
line = ""; // ล้างค่าตัวแปร line

break; // ออกจากลูป

} else if (c == '\n') { // ถ้าเป็น \n (new line)

// Pass {new line}

} else { // ถ้าไม่ใช่

line += c; // เพิ่มข้อมูล 1 ไบต์ ไปต่อท้ายในตัวแปร line

}

}

}



delay(1);

client.stop(); // ปิดการเชื่อมต่อกับ Client

Serial.println("Client disconnect"); // ส่งข้อความว่า Client disconnect ไปที่ Serial Monitor

}
```

ESP8266_TCPServerLED.ino hosted with ❤ by GitHub view raw

เชื่อมต่อเข้าไปที่ ESP8266 แล้วใช้โปรแกรม hercules ทดลองส่ง LEDON<LF><CR> และ LEDOFF<LF><CR> จะพบว่าสามารถควบคุมหลอด LED ได้แล้ว



ส่งท้าย

บทความนี้เป็นเพียงตัวอย่างเล็ก ๆ ในขณะที่ ESP8266 มีความสามารถเยอะมากกว่านี้ครับ หวังว่าบทความนี้จะช่วยให้ท่านสามารถเรียนรู้และนำไปประยุกต์ใช้งานได้ครับ

ขอบคุณที่ติดตามอ่านมาถึงส่วนนี้ครับ