

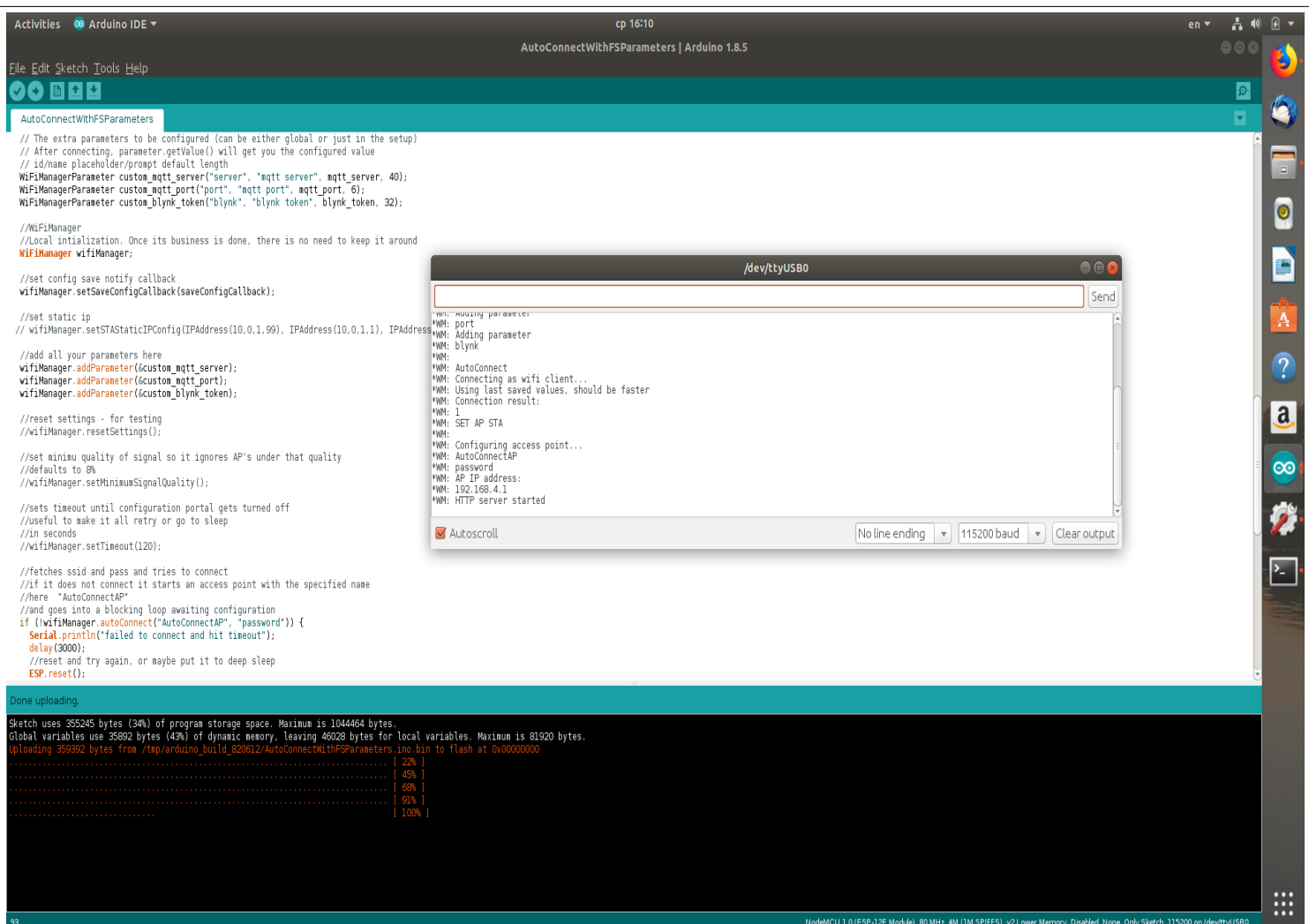
## ESP8266 login procedure example

Step 1. Install this libraries: WifiManager and ArduinoJson. Click – Sketch → Include library → Manage libraries and type name of the library. In this project we use WifiManager library → WhareHauoraWifiManager and standard ArduinoJson library.

Step 2. Clone github repo from this link: [https://github.com/ihor96berizka/Smart\\_house\\_ESP8266\\_NodeMCU](https://github.com/ihor96berizka/Smart_house_ESP8266_NodeMCU). Go to the AutoConnectWithFSPParameters folder. Here you will find Arduino project which performs login procedure.

Step 3. Connect your ESP8266 NodeMCU module to PC. Open this project in Arduino IDE and program MCU.

Step 4. Open Serial Monitor, set baudrate = 115200. If everything is correct – you`ll see output like on figure 1.



The screenshot displays the Arduino IDE interface. The main window shows the sketch 'AutoConnectWithFSPParameters' with the following code:

```
// The extra parameters to be configured (can be either global or just in the setup)
// After connecting, parameter.getValue() will get you the configured value
// id/name placeholder/prompt default length
WifiManagerParameter custom_mqtt_server("server", "mqtt server", mqtt_server, 40);
WifiManagerParameter custom_mqtt_port("port", "mqtt port", mqtt_port, 6);
WifiManagerParameter custom_blynk_token("blynk", "blynk token", blynk_token, 32);

//WifiManager
//Local initialization. Once its business is done, there is no need to keep it around
WifiManager wifiManager;

//set config save notify callback
wifiManager.setSaveConfigCallback(saveConfigCallback);

//set static ip
// wifiManager.setStaticIPConfig(IPAddress(10,0,1,99), IPAddress(10,0,1,1), IPAddress(10,0,1,1));

//add all your parameters here
wifiManager.addParameter(custom_mqtt_server);
wifiManager.addParameter(custom_mqtt_port);
wifiManager.addParameter(custom_blynk_token);

//reset settings - for testing
//wifiManager.resetSettings();

//set minimum quality of signal so it ignores AP's under that quality
//defaults to 8%
//wifiManager.setMinimumSignalQuality();

//sets timeout until configuration portal gets turned off
//useful to make it all retry or go to sleep
//in seconds
//wifiManager.setTimeout(120);

//fetches ssid and pass and tries to connect
//if it does not connect it starts an access point with the specified name
//here "AutoConnectAP"
//and goes into a blocking loop awaiting configuration
if (!wifiManager.autoConnect("AutoConnectAP", "password")) {
  Serial.println("failed to connect and hit timeout");
  delay(3000);
  //reset and try again, or maybe put it to deep sleep
  ESP.reset();
}
```

The Serial Monitor window, titled '/dev/ttyUSB0', shows the following output:

```
WiFi: adding parameter
WiFi: port
WiFi: Adding parameter
WiFi: blynk
WiFi:
WiFi: AutoConnect
WiFi: Connecting as wifi client...
WiFi: Using last saved values, should be faster
WiFi: Connection result:
WiFi: 1
WiFi: SET AP STA
WiFi:
WiFi: Configuring access point...
WiFi: AutoConnectAP
WiFi: password
WiFi: AP IP address:
WiFi: 192.168.4.1
WiFi: HTTP server started
```

Below the Serial Monitor, the IDE status bar indicates 'Done uploading.' and provides memory usage statistics:

```
Sketch uses 355245 bytes (34%) of program storage space. Maximum is 1044464 bytes.
Global variables use 35892 bytes (43%) of dynamic memory, leaving 46028 bytes for local variables. Maximum is 81920 bytes.
Uploading 359392 bytes from /tmp/arduino_build_820612/AutoConnectWithFSPParameters.ino.bin to flash at 0x00000000
```

The bottom status bar shows 'NodeMCU 1.0 (ESP-12E Module, 80 MHz, 4M (1M SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/ttyUSB0)'.

Figure 1. Output in Serial Monitor after ESP8266 module restart.

As you can see from output: ESP8266 now working in access point mode and hosts server with default address: 192.168.4.1 so you can turn on any device with wi-fi and connect to ESP8266. SSID = ESP\_AP, password = password.

Step 5. Connect to this access point using smartphone\PC\laptop. I have used RPi3 for this example. Open your browser and in address textbox write address of ESP8266 server: 192.168.4.1. Then you will see window like on figure 2.

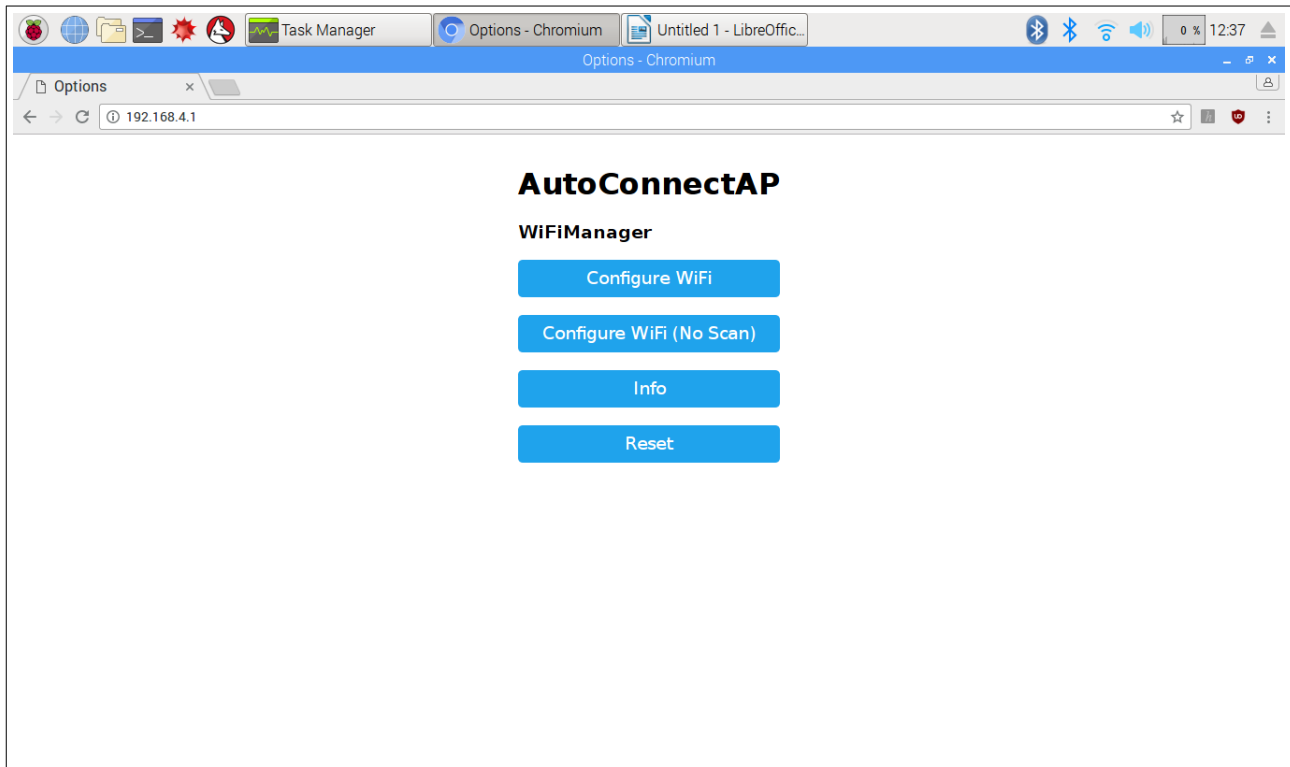


Figure 2. Login menu

Step 6. Click on Configure WiFi button. You will see window like on figure 3.

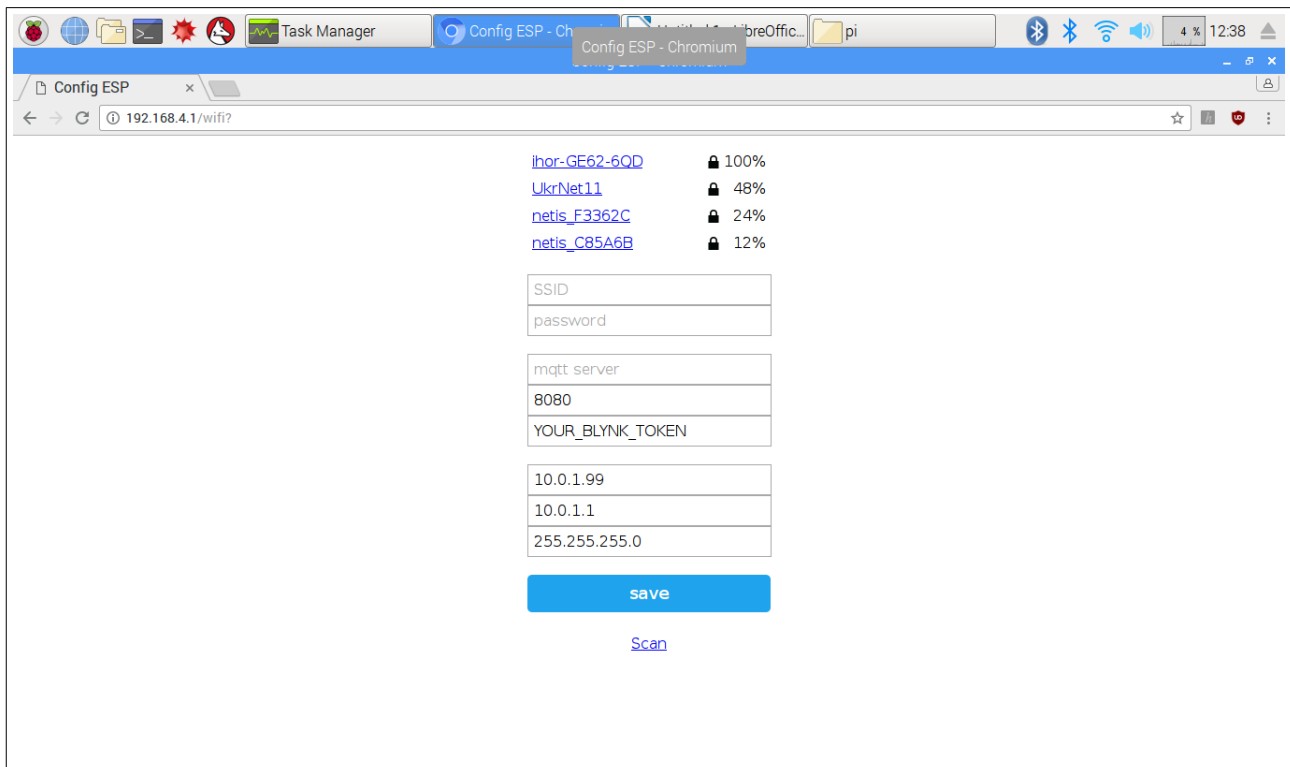


Figure 3. Configure WiFi window.

Here you can see all available WiFi access points. Choose one of them (for example first one) and enter password.

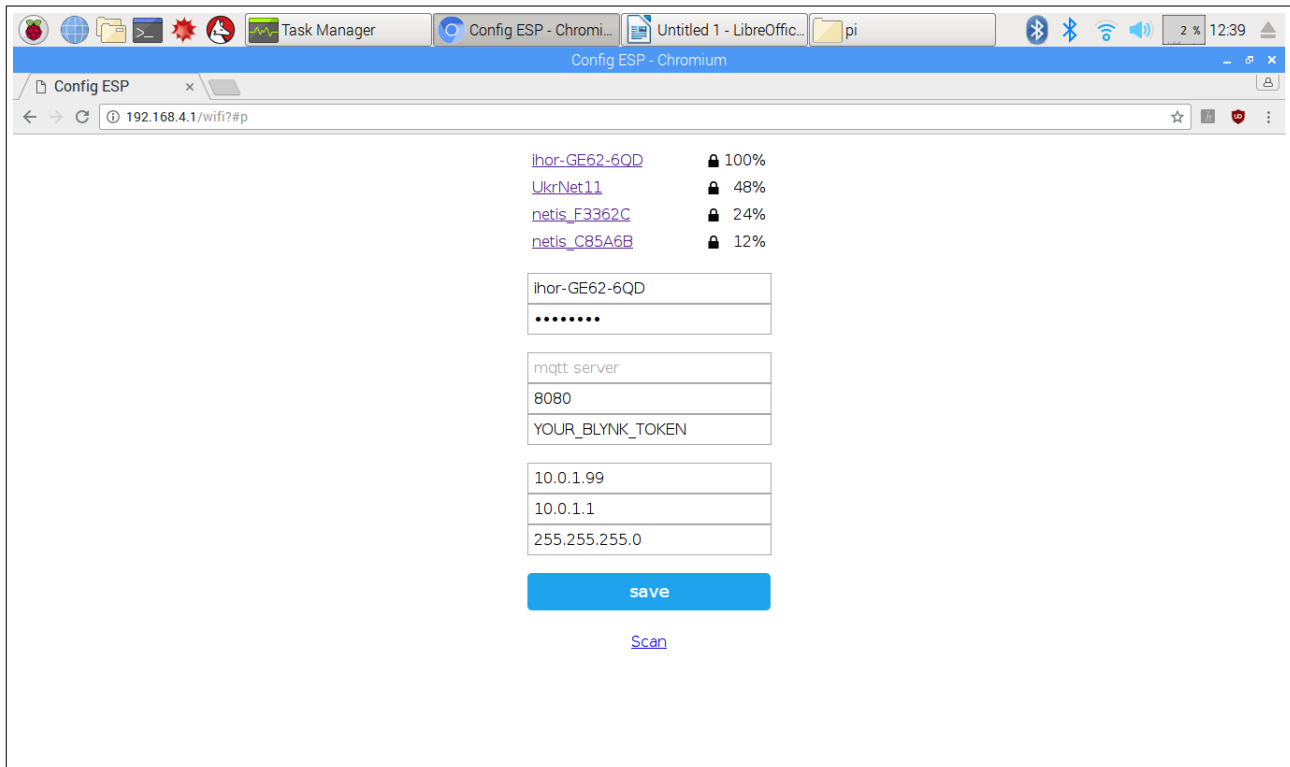


Figure 4. Configuring WiFi connection.

Next press save button. You will see window like on figure 5.

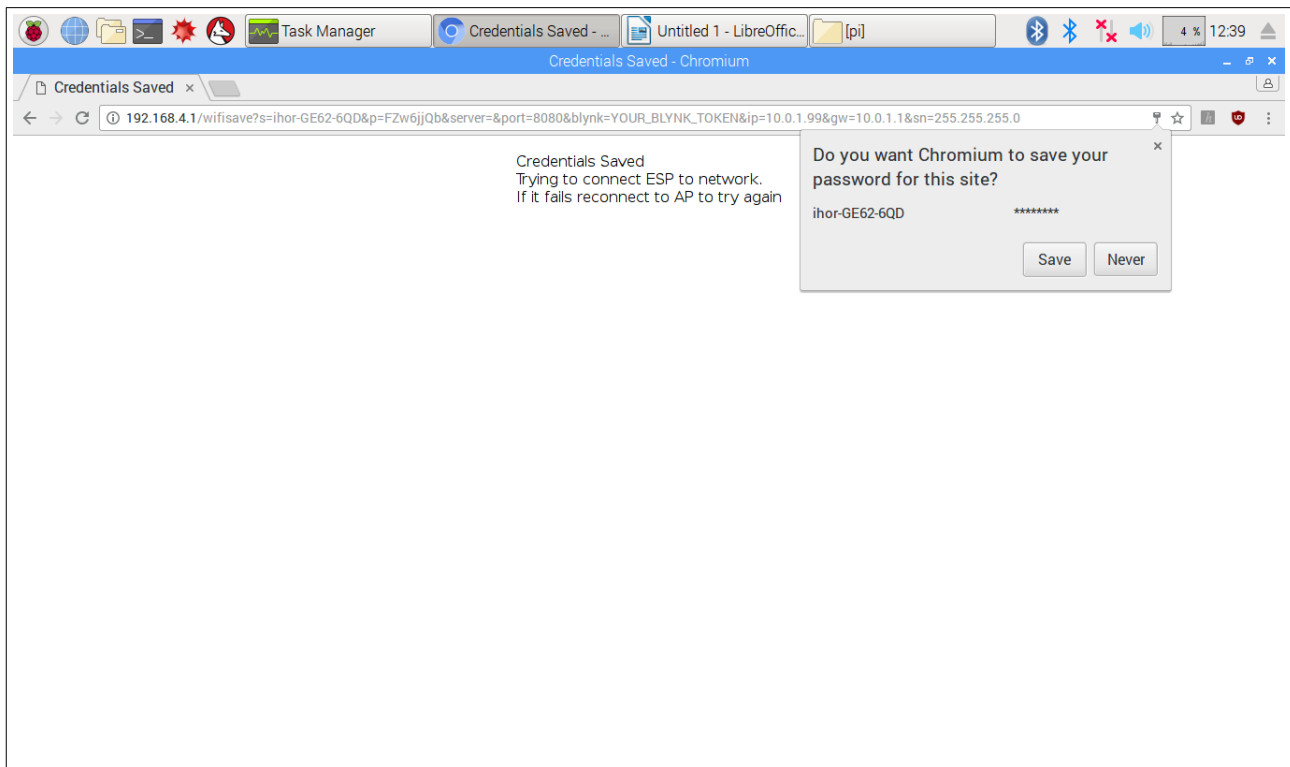
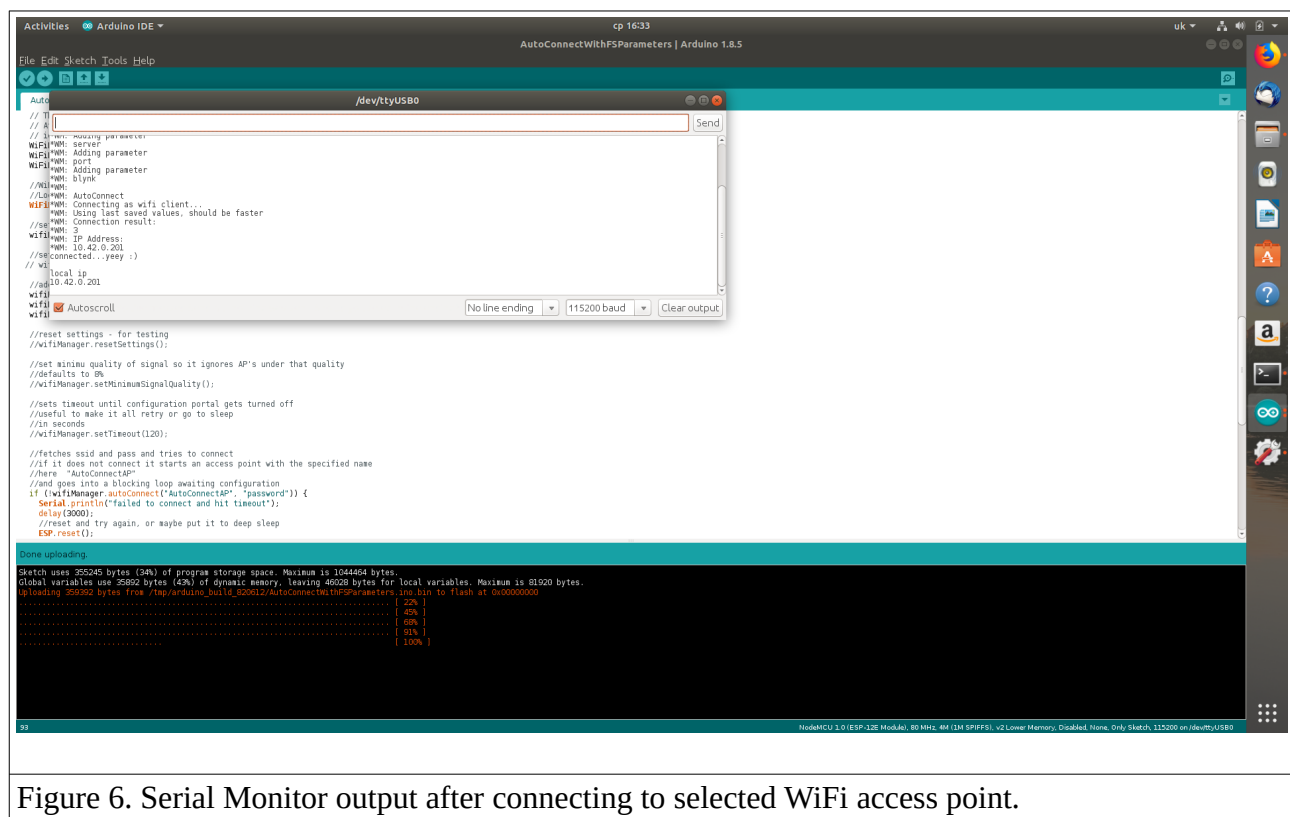


Figure 5. Finishing configuring WiFi connection.

Step. 7. To verify if ESP8266 module is connected to selected WiFi access point – open Serial Monitor. If everything went right – you will see output like on figure below.



On figure 6 you can see message that module is connected to network and its local IP address.

Step 8. Now its time to verify that module is really connected to specified WiFi access point.

If your access point is router – open router menu and check if this IP address is in list of connected devices.

In this example I have used my laptop as WiFi hotspot. So, open terminal on your laptop and enter command: `arp -e`. This command will list all connected devices to this access point and their IP addresses. Output is show on figure below. As you can see – there is one device connected to this WiFi access point (wlp2s0 interface) and it has IP addresses similar to our ESP8266 module. So, we can assume that this is our ESP8266 module.

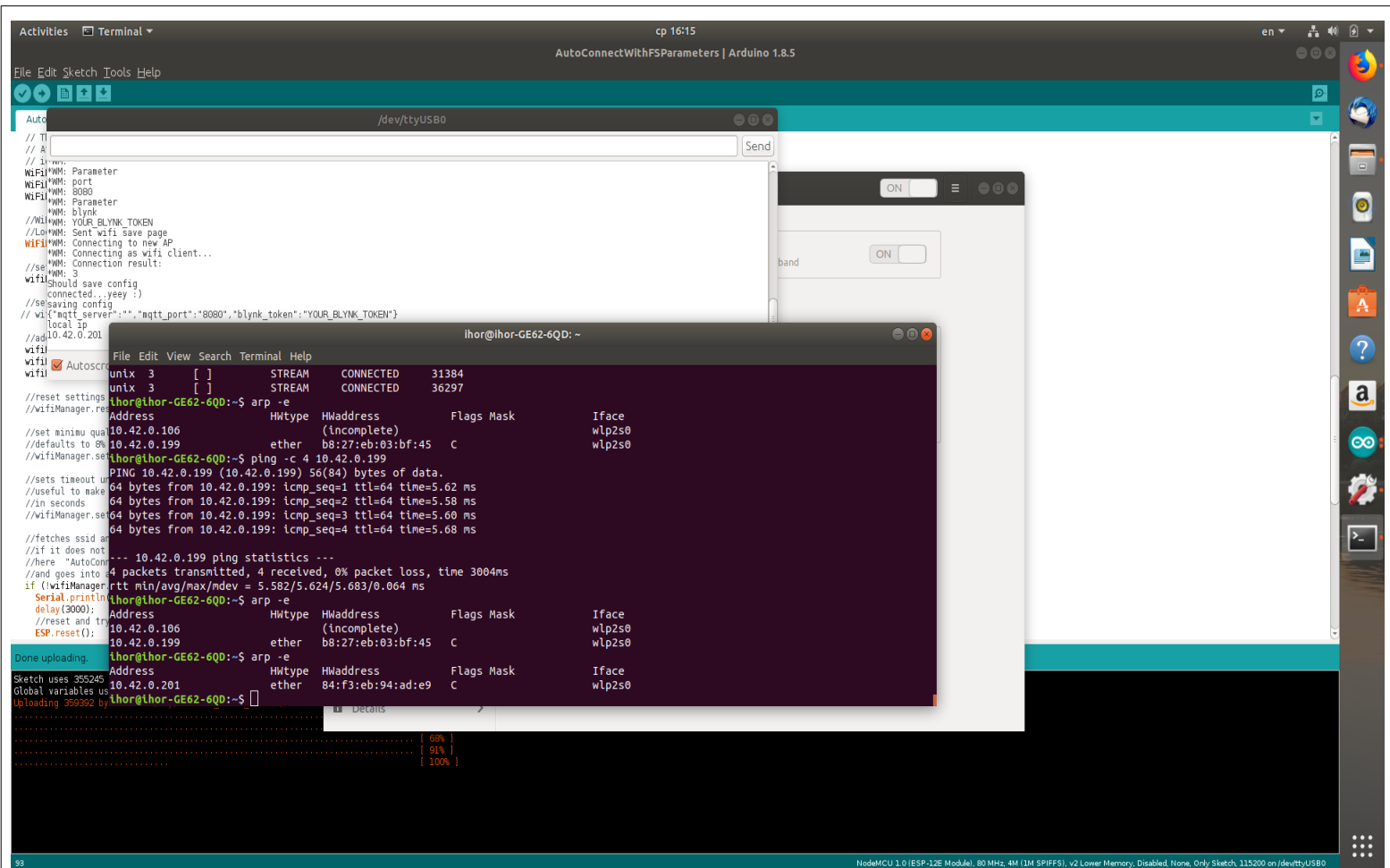


Figure 7. Output of arp -e command.