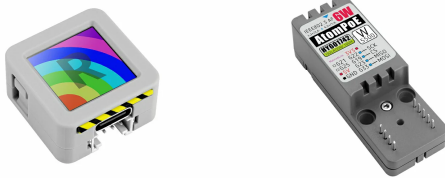


Atomic PoE Base Arduino Tutorial

1. Preparation

- Environment setup: Refer to the [Arduino IDE Getting Started Guide](#) to complete IDE installation, and install the corresponding board package and required libraries according to your development board.
- Required libraries:
 - [M5Unified](#)
 - [M5GFX](#)
 - [M5-Ethernet](#)
 - [PubSubClient](#)
 - [ArduinoHttpClient](#)
- Required hardware:
 - [AtomS3R](#)
 - [Atomic PoE Base](#)



2. Example

- The main controller used in this tutorial is AtomS3R, paired with Atomic PoE Base. Atomic PoE Base communicates with the host via SPI. Modify the pin definitions in the program according to your actual circuit connections. After connection, the corresponding SPI IOs are G5 (SCK)、G7 (MISO)、G8 (MOSI)、G6 (CS).

2.1 Web Server

```
1  #include "M5Unified.h"
2  #include "SPI.h"
3  #include "M5_Ethernet.h"
4
5  #define SCK 5
6  #define MISO 7
7  #define MOSI 8
8  #define CS 6
9
10 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x99}; // Custom MAC address
11 // IPAddress ip(192, 168, 1, 177); // Static IP (Custom)
12
13 EthernetServer server(80); // Initialize the Web Server on port 80 (standard HTTP port)
14
15 void setup() {
16     M5.begin();
17     Serial.begin(115200);
18
19     M5.Display.setTextDatum(middle_center);
20     M5.Display.setFont(&fonts::Font2);
21
22     SPI.begin(SCK, MISO, MOSI, -1);
23     Ethernet.init(CS);
24
25     M5.Display.drawString("Init...", M5.Display.width() / 2, M5.Display.height() / 2);
26     Serial.println("Initializing...");
27     delay(500);
28
29     // To use a static IP, use Ethernet.begin(mac, ip)
30     // Attempt to obtain an IP address via DHCP
31     while (Ethernet.begin(mac) != 1) {
32         Serial.println("Error getting IP address via DHCP, trying again...");
33         delay(1000);
34     }
35
36     // Check for Ethernet hardware presence
37     if (Ethernet.hardwareStatus() == EthernetNoHardware) {
38         Serial.println(
39             "Ethernet shield was not found. Sorry, can't run without "
40             "hardware. :(");
41         while (true) {
42             delay(1); // Do nothing if hardware is missing
43         }
44     }
45     if (Ethernet.linkStatus() == LinkOFF) {
46         Serial.println("Ethernet cable is not connected.");
47     }
48
49     // Start web server
50     server.begin();
```

```

51     Serial.print("Server is at ");
52     Serial.println(Ethernet.localIP());
53     M5.Display.clear();
54     M5.Display.drawString(Ethernet.localIP().toString().c_str(),
55                           M5.Display.width() / 2, M5.Display.height() / 2);
56     delay(1000);
57 }
58
59 long lastTime;
60
61 void loop() {
62     // Listen for incoming client connections
63     EthernetClient client = server.available();
64     if (client) {
65         Serial.println("new client");
66
67         boolean currentLineIsBlank = true; // HTTP requests terminate with a blank line
68
69         while (client.connected()) {
70             if (client.available()) {
71                 char c = client.read();
72                 Serial.write(c); // Echo request to Serial Monitor for debugging
73                 // If you've gotten to the end of the line (received a newline
74                 // character) and the line is blank, the http request has ended,
75                 // so you can send a reply
76                 if (c == '\n' && currentLineIsBlank) {
77                     // Send a standard http response header
78                     client.println("HTTP/1.1 200 OK");
79                     client.println("Content-Type: text/html");
80                     client.println(
81                         "Connection: close"); // Connection will be closed
82                                             // after completion of the
83                                             // response
84                     client.println("Refresh: 5"); // Refresh the page
85                                                  // automatically every 5 sec
86
87                     client.println();
88                     client.println("<!DOCTYPE HTML>");
89                     client.println("<html>");
90                     client.print("<h2>Hello M5Stack User!</h2>");
91                     client.println("</html>");
92                     break;
93                 }
94                 if (c == '\n') {
95                     // Starting a new line
96                     currentLineIsBlank = true;
97                 } else if (c != '\r') {
98                     // You've gotten a character on the current line
99                     currentLineIsBlank = false;
100                 }
101             }
102             // Give the web browser time to receive the data
103             delay(1);

```

```
104         // Close the connection:
105         client.stop();
106         Serial.println("client disconnected");
107     }
108 }
```

| 2.2 MQTT

```
1  #include "M5Unified.h"
2  #include "SPI.h"
3  #include "M5_Ethernet.h"
4  #include "PubSubClient.h"
5
6  #define SCK 5
7  #define MISO 7
8  #define MOSI 8
9  #define CS 6
10
11 #define PUB_INTERVAL 3000 // Publishing interval unit: ms
12 #define PUB_TOPIC "Atomic_PoE_Send"
13 #define SUB_TOPIC "Atomic_PoE_Receive"
14
15 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x99}; // Custom MAC address
16 // IPAddress ip(192, 168, 1, 177); // Static IP (Custom)
17
18 const char* mqtt_server = "mqtt.m5stack.com"; // MQTT broker address
19 EthernetClient ethClient;
20 PubSubClient client(ethClient);
21
22 // Handle incoming MQTT messages
23 void callback(char* topic, byte* payload, unsigned int length) {
24     Serial.print("Message arrived [");
25     Serial.print(topic);
26     Serial.print("] ");
27
28     String payloadStr;
29     for (int i = 0; i < length; i++) {
30         payloadStr += (char)payload[i];
31     }
32     Serial.print(payloadStr);
33     M5.Display.fillRect(0, 0, 240, 68, TFT_BLACK);
34     M5.Display.drawString("Rece: " + payloadStr, M5.Display.width() / 2, 30);
35
36 }
37
38 // Connect or reconnect to the MQTT broker
39 void reconnect() {
40     // Loop until we're reconnected
41     while (!client.connected()) {
42         Serial.print("Attempting MQTT connection...\n");
43         // Attempt to connect
44         if (client.connect("arduinoClient")) {
45             Serial.println("Connected");
46             M5.Display.clear();
47             M5.Display.drawString("Connected!", M5.Display.width() / 2, M5.Display.height() / 2);
48
49             // Once connected, publish an announcement...
50             client.publish(PUB_TOPIC, "Client connected");
```

```

51         // ... and resubscribe
52         client.subscribe(SUB_TOPIC);
53     } else {
54         M5.Display.clear();
55         M5.Display.drawString("Failed!", M5.Display.width() / 2, 60);
56         Serial.print("Failed, rc=");
57         Serial.print(client.state());
58         Serial.println(" try again in 5s");
59         // Wait 5 seconds before retrying
60         delay(5000);
61     }
62 }
63 }
64
65 void setup() {
66     M5.begin();
67     Serial.begin(115200);
68
69     M5.Display.setTextColor(GREEN);
70     M5.Display.setTextDatum(middle_center);
71     M5.Display.setFont(&fonts::Font2);
72
73     SPI.begin(SCK, MISO, MOSI, -1);
74     Ethernet.init(CS);
75
76     M5.Display.drawString("Init...", M5.Display.width() / 2, M5.Display.height() / 2);
77     Serial.println("Initializing...");
78
79     // To use a static IP, use Ethernet.begin(mac, ip)
80     // Attempt to obtain an IP address via DHCP
81     while (Ethernet.begin(mac) != 1) {
82         Serial.println("Error getting IP address via DHCP, trying again...");
83         delay(1000);
84     }
85
86     // Check for Ethernet hardware presence
87     if (Ethernet.hardwareStatus() == EthernetNoHardware) {
88         Serial.println(
89             "Ethernet shield was not found. Sorry, can't run without :(");
90         while (true) {
91             delay(1); // Do nothing if hardware is missing
92         }
93     }
94     if (Ethernet.linkStatus() == LinkOFF) {
95         Serial.println("Ethernet cable is not connected.");
96     }
97
98     Serial.print("IP Address: ");
99     Serial.println(Ethernet.localIP());
100
101     M5.Display.clear();
102     M5.Display.drawString(Ethernet.localIP().toString().c_str(),
103         M5.Display.width() / 2, M5.Display.height() / 2);

```

```

104 // --- MQTT Setup ---
105 client.setServer(mqtt_server, 1883);
106 client.setCallback(callback); // Register incoming message handler
107 delay(1000);
108 }
109
110 long lastTime;
111
112 void loop() {
113     if (!client.connected()) {
114         reconnect();
115         delay(500);
116         M5.Display.clear();
117         M5.Display.drawString("Rece: NULL", M5.Display.width() / 2, 30);
118     } else {
119         client.loop(); // Essential to process incoming messages and maintain the heartbeat
120         if (millis() - lastTime > PUB_INTERVAL) {
121             lastTime = millis();
122             M5.Display.fillRect(0, 68, 240, 67, TFT_BLACK);
123             M5.Display.drawString("Send: " + String(lastTime) + " ms", M5.Display.width() / 2, 100);
124             String data = "Hello world: " + String(lastTime);
125             client.publish(PUB_TOPIC, data.c_str());
126             Serial.println("Send Message [" PUB_TOPIC "] " + data);
127         }
128     }
129 }

```

2.3 HTTP

```
1  #include "M5Unified.h"
2  #include "SPI.h"
3  #include "M5_Ethernet.h"
4  #include "ArduinoHttpClient.h"
5
6  #define SCK 5
7  #define MISO 7
8  #define MOSI 8
9  #define CS 6
10
11 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x99}; // Custom MAC address
12 // IPAddress ip(192, 168, 1, 177); // Static IP (Custom)
13
14 const char* http_server = "httpbin.org"; // Target HTTP server
15 EthernetClient ethClient;
16 HttpClient client = HttpClient(ethClient, http_server);
17
18 void setup() {
19     M5.begin();
20     Serial.begin(115200);
21
22     M5.Display.setTextColor(GREEN);
23     M5.Display.setTextDatum(middle_center);
24     M5.Display.setFont(&fonts::Font2);
25
26     SPI.begin(SCK, MISO, MOSI, -1);
27     Ethernet.init(CS);
28
29     M5.Display.drawString("Init...", M5.Display.width() / 2, 60);
30     Serial.println("Initializing...");
31
32     // To use a static IP, use Ethernet.begin(mac, ip)
33     // Attempt to obtain an IP address via DHCP
34     while (Ethernet.begin(mac) != 1) {
35         Serial.println("Error getting IP address via DHCP, trying again...");
36         delay(1000);
37     }
38
39     // Check for Ethernet hardware presence
40     if (Ethernet.hardwareStatus() == EthernetNoHardware) {
41         Serial.println(
42             "Ethernet shield was not found. Sorry, can't run without hardware. :(");
43         while (true) {
44             delay(1); // Do nothing if hardware is missing
45         }
46     }
47     if (Ethernet.linkStatus() == LinkOFF) {
48         Serial.println("Ethernet cable is not connected.");
49     }
50 }
```

```

51     Serial.print("IP Address: ");
52     Serial.println(Ethernet.localIP());
53
54     M5.Display.clear();
55     M5.Display.drawString(Ethernet.localIP().toString().c_str(),
56                           M5.Display.width() / 2, M5.Display.height() / 2);
57     delay(1000);
58 }
59
60 void loop() {
61     // --- GET request ---
62     M5.Display.clear();
63     M5.Display.drawString("GET", M5.Display.width() / 2, 20);
64     Serial.println("making GET request");
65
66     client.get("/get");
67     // read the status code and body of the response
68     int statusCode = client.responseStatusCode();
69     String response = client.responseBody();
70
71     Serial.print("Status code: ");
72     Serial.println(statusCode);
73     Serial.print("Response: ");
74     Serial.println(response);
75     Serial.println("Wait five seconds");
76
77     M5.Display.drawString("STATUS:", M5.Display.width() / 2, 60);
78     M5.Display.drawString(String(statusCode), M5.Display.width() / 2, 100);
79
80     delay(5000);
81
82     // --- POST request ---
83     M5.Display.clear();
84     M5.Display.drawString("POST", M5.Display.width() / 2, 20);
85     Serial.println("making POST request");
86
87     String contentType = "application/x-www-form-urlencoded";
88     String postData    = "name=Alice&age=12";
89
90     client.post("/post", contentType, postData);
91
92     // read the status code and body of the response
93     statusCode = client.responseStatusCode();
94     response   = client.responseBody();
95
96     Serial.print("Status code: ");
97     Serial.println(statusCode);
98     Serial.print("Response: ");
99     Serial.println(response);
100    Serial.println("Wait five seconds");
101
102    M5.Display.drawString("STATUS:", M5.Display.width() / 2, 60);
103    M5.Display.drawString(String(statusCode), M5.Display.width() / 2, 100);

```

```
104  
105     delay(5000);  
106 }
```

2.4 NTP

```
1  #include "M5Unified.h"
2  #include "SPI.h"
3  #include "M5_Ethernet.h"
4
5  #define SCK 5
6  #define MISO 7
7  #define MOSI 8
8  #define CS 6
9
10 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x99}; // Custom MAC address
11 // IPAddress ip(192, 168, 1, 177); // Static IP (Custom)
12
13 const char* ntpServerName = "cn.pool.ntp.org"; // NTP Server address
14 const uint16_t localPort = 8888; // Local port to listen for UDP packets
15 const int timeZone = 8; // Time zone offset (Beijing is UTC+8)
16 const int NTP_PACKET_SIZE = 48; // NTP packet size
17
18 byte packetBuffer[NTP_PACKET_SIZE]; // Buffer for incoming and outgoing packets
19 EthernetUDP Udp;
20
21 void sendNTPpacket(const char* address);
22 void printFormattedTime(uint32_t rawTime);
23
24 void setup() {
25     M5.begin();
26     Serial.begin(115200);
27
28     M5.Display.setTextColor(GREEN);
29     M5.Display.setTextDatum(middle_center);
30     M5.Display.setFont(&fonts::Font2);
31
32     SPI.begin(SCK, MISO, MOSI, -1);
33     Ethernet.init(CS);
34
35     M5.Display.drawString("Init...", M5.Display.width() / 2, M5.Display.height() / 2);
36     Serial.println("Initializing...");
37
38     // To use a static IP, use Ethernet.begin(mac, ip)
39     // Attempt to obtain an IP address via DHCP
40     while (Ethernet.begin(mac) != 1) {
41         Serial.println("Error getting IP address via DHCP, trying again...");
42         delay(1000);
43     }
44
45     // Check for Ethernet hardware presence
46     if (Ethernet.hardwareStatus() == EthernetNoHardware) {
47         Serial.println(
48             "Ethernet shield was not found. Sorry, can't run without hardware. :(");
49         while (true) {
50             delay(1); // Do nothing if hardware is missing
```

```

51     }
52 }
53 if (Ethernet.linkStatus() == LinkOFF) {
54     Serial.println("Ethernet cable is not connected.");
55 }
56
57 Serial.print("IP Address: ");
58 Serial.println(Ethernet.localIP());
59
60 M5.Display.clear();
61 M5.Display.drawString(Ethernet.localIP().toString().c_str(),
62                       M5.Display.width() / 2, M5.Display.height() / 2);
63 M5.Display.setTextDatum(top_left);
64
65 Udp.begin(localPort); // Start listening for UDP packets
66 delay(1000);
67 }
68
69 void loop() {
70     Serial.println("Sending NTP packet...");
71     sendNTPpacket(ntpServerName);
72
73     delay(1000); // Wait for packet to return
74
75     if (Udp.parsePacket()) { // Check if a response is received
76         Serial.println("Packet received");
77         Udp.read(packetBuffer, NTP_PACKET_SIZE);
78
79         // --- Parse NTP Timestamp ---
80         // NTP timestamp is located at bytes 40-43, representing seconds since Jan 1, 1900
81         uint32_t highWord = word(packetBuffer[40], packetBuffer[41]);
82         uint32_t lowWord = word(packetBuffer[42], packetBuffer[43]);
83         uint32_t secsSince1900 = highWord << 16 | lowWord; // Combine into a 32-bit unsigned integer
84
85         // --- Convert to Unix Epoch Time ---
86         // Unix epoch starts in 1970; NTP starts in 1900.
87         // The 70-year difference (including leap years) is 2,208,988,800 seconds.
88         const uint32_t seventyYears = 2208988800UL;
89         uint32_t epoch = secsSince1900 - seventyYears;
90
91         uint32_t bjTime = epoch + (timeZone * 3600); // Adjust for Beijing time zone
92         printFormattedTime(bjTime);
93     } else {
94         Serial.println("No packet received yet.");
95     }
96
97     delay(4000); // Sync every 5 seconds, 4+1
98 }
99
100 // Send an NTP request to the given address
101 void sendNTPpacket(const char* address) {
102     memset(packetBuffer, 0, NTP_PACKET_SIZE);
103     // Set the NTP request header

```

```

104 // 0b00011011 represents: LI=00=0 (no warning), VN=011=3 (Version 3), Mode=011=3 (Client mode)
105 packetBuffer[0] = 0b00011011;
106
107 Udp.beginPacket(address, 123); // NTP requests are on port 123
108 Udp.write(packetBuffer, NTP_PACKET_SIZE);
109 Udp.endPacket();
110 }
111
112 // Format and display the time
113 void printFormattedTime(uint32_t rawTime) {
114     int hours = (rawTime % 86400L) / 3600; // 86400 seconds = 1 day
115     int minutes = (rawTime % 3600) / 60;
116     int seconds = rawTime % 60;
117
118     String timeStr = "Time: " + String(hours) + ":" +
119                     (minutes < 10 ? "0" : "") + String(minutes) + ":" +
120                     (seconds < 10 ? "0" : "") + String(seconds);
121
122     Serial.println(timeStr);
123
124     M5.Display.clear();
125     M5.Display.drawString("Beijing", 10, 30);
126     M5.Display.drawString(timeStr, 10, 60);
127 }

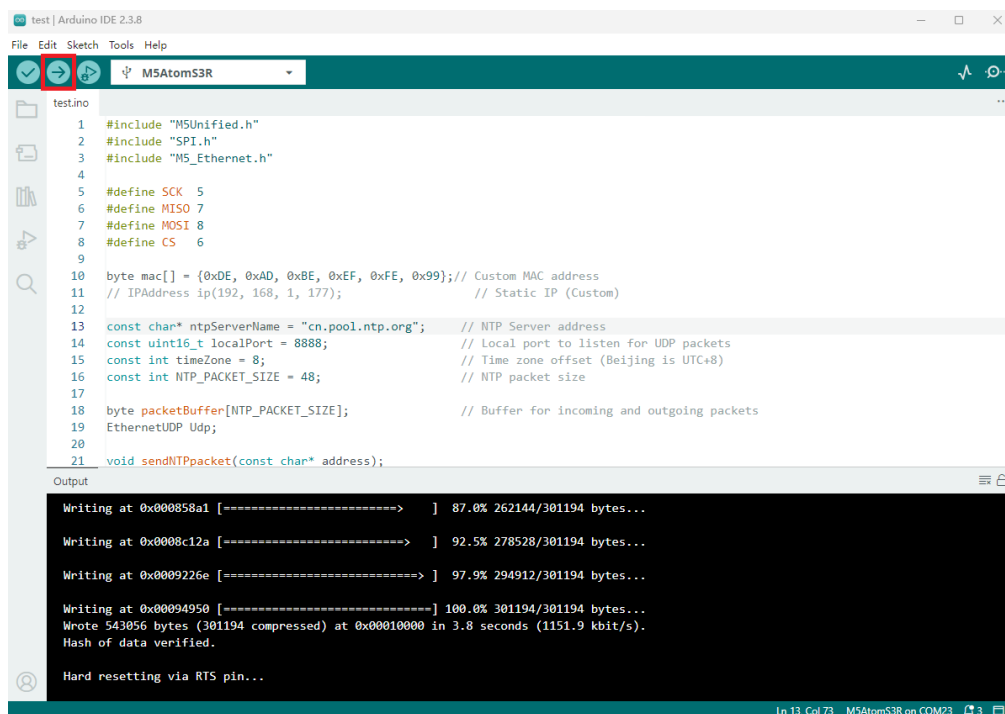
```

3. Compile & Upload

- 1. Press and hold the Boot button on AtomS3R, then release it when the green LED turns on. The device will enter download mode and wait for programming.



- 2. Select the device port and click the compile and upload button in the upper left corner of Arduino IDE. Wait for the program to complete compilation and upload to the device.



```
testino
1 #include "M5Unified.h"
2 #include "SPI.h"
3 #include "M5_Ethernet.h"
4
5 #define SCK 5
6 #define MISO 7
7 #define MOSI 8
8 #define CS 6
9
10 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x99}; // Custom MAC address
11 // IPAddress ip(192, 168, 1, 177); // Static IP (Custom)
12
13 const char* ntpServerName = "cn.pool.ntp.org"; // NTP Server address
14 const uint16_t localPort = 8888; // Local port to listen for UDP packets
15 const int timeZone = 8; // Time zone offset (Beijing is UTC+8)
16 const int NTP_PACKET_SIZE = 48; // NTP packet size
17
18 byte packetBuffer[NTP_PACKET_SIZE]; // Buffer for incoming and outgoing packets
19 EthernetUDP Udp;
20
21 void sendNTPpacket(const char* address);
```

Output

```
Writing at 0x000858a1 [=====] 87.0% 262144/301194 bytes...
Writing at 0x0008c12a [=====] 92.5% 278528/301194 bytes...
Writing at 0x0009226e [=====] 97.9% 294912/301194 bytes...
Writing at 0x00094950 [=====] 100.0% 301194/301194 bytes...
Wrote 543056 bytes (301194 compressed) at 0x00010000 in 3.8 seconds (1151.9 kbit/s).
Hash of data verified.
Hard resetting via RTS pin...
```

4. Network connection effect

- Web Server

After power-on, Host machine will automatically initialize the Atomic PoE Base module and create a web server. The server IP address will be displayed on the screen. Enter this IP address in your browser to see the message **Hello M5 User!** on the page, and the page will refresh automatically every 5 seconds.



Serial monitor feedback is as follows:

```
Initializing...
Server is at 192.168.20.157
new client
GET / HTTP/1.1
Host: 192.168.20.157
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6,ja;q=0.5,ko;q=0.4

client disconnected
```

- MQTT

After power-on, Host machine will automatically configure the Atomic PoE Base module and connect to the MQTT broker. Once connected, it publishes a message to the broker every 3 seconds. The screen shows send/receive information, and the serial monitor provides more details.



Serial monitor feedback is as follows:

```
Initializing...
IP Address: 192.168.20.157
Attempting MQTT connection...
Connected
Send Message [Atomic_PoE_Send] Hello world: 3001
Send Message [Atomic_PoE_Send] Hello world: 6002
Send Message [Atomic_PoE_Send] Hello world: 9003
Send Message [Atomic_PoE_Send] Hello world: 12004
Send Message [Atomic_PoE_Send] Hello world: 15005
Send Message [Atomic_PoE_Send] Hello world: 18006
Send Message [Atomic_PoE_Send] Hello world: 21007
Send Message [Atomic_PoE_Send] Hello world: 24008
Message arrived [Atomic_PoE_Receive] Hello!
...
```

- HTTP

After power-on, Host machine will automatically configure the Atomic PoE Base module and connect to the HTTP server, then continuously send GET and POST requests in a loop. The main screen displays the request status, and the serial monitor provides more details.



Serial monitor feedback is as follows:

```
Initializing...
IP Address: 192.168.20.157
making GET request
Status code: 200
Response: {
  "args": {},
  "headers": {
    "Host": "httpbin.org",
    "User-Agent": "Arduino/2.2.0",
    "X-Amzn-Trace-Id": "Root=1-69524204-7d767e3110e3bd6d1e28f6a8"
  },
  "origin": "113.88.164.214",
  "url": "http://httpbin.org/get"
}
```

```
Wait five seconds
making POST request
Status code: 200
Response: {
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "age": "12",
    "name": "Alice"
  },
  "headers": {
    "Content-Length": "17",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "Arduino/2.2.0",
    "X-Amzn-Trace-Id": "Root=1-6952420a-588b0f181a6d7b311e694e06"
  },
  "json": null,
  "origin": "113.88.164.214",
  "url": "http://httpbin.org/post"
}
```

- o NTP

After power-on, Host machine will automatically configure the Atomic PoE Base module and connect to the NTP server, synchronizing the time once every 5 seconds.



Serial monitor feedback is as follows:

```
Initializing...
IP Address: 192.168.20.157
Sending NTP packet...
No packet received yet.
Sending NTP packet...
Packet received
Time: 17:49:00
Sending NTP packet...
Packet received
Time: 17:49:05
```