

1. Preparations

- Environment Configuration: Refer to [Arduino IDE Getting Started Tutorial](#) to complete IDE installation, and install corresponding board manager and required driver libraries according to the development board being used.
- Required Driver Libraries:
 - [M5AtomS3](#)
 - [ATOM_DTU_NB](#)
 - [TinyGsmClient](#)
 - [ArduinoHttpClient](#)

Note

You need to download the latest library version from GitHub: [TinyGsmClient - M5Stack GitHub](#). Do not download from Arduino Library. (For any questions, please refer to [this tutorial](#))

- Hardware Products Used:
 - [AtomS3-Lite](#)
 - [Atom DTU NBloT2](#)



2. Notes

SIM Card Compatibility

This module requires a Micro SIM card to access network services normally. Modern IoT SIM cards support both Cat-M1 and NB-IOT, or you can choose traditional SIM cards that support both services. Please select cards certified by the following carriers: Deutsche Telekom / Vodafone / Telefonica / China Telecom / China Mobile / China Unicom.

Note

Do not use the SIM card in different devices, as this may cause the SIM card to be locked.

Pin Compatibility

Since each host has different pin configurations, M5Stack officially provides a [Pin Compatibility Table](#) for user convenience. Please modify the sample program according to the actual pin connections.



Atom DTU NB IoT2

AtomS3R

AtomS3R

AtomS3R-CAM

AtomS3R-M12

AtomS3R-Ext

AtomS3

AtomS3-Lite

Atom-Lite

Atom-Matrix

Atom-Echo

G7

G8

5V

GND

Fixed Connect

Switch Connect

Atomic-BUS

UART_TX

UART_RX

RS485_TX

RS485_RX

PORTA

PORTA

5V

GND

"Atom DTU NB IoT2" & "AtomS3R" Bus Connection Note:

Please modify the example program according to the actual PIN number when using.

Fixed Connect: Fixed connection, cannot be changed.

Switch Connect: Supports switching pin connections via jumpers or DIP switches.

3. HTTP Service

By default, it uses the CAT mode of the SIM card. To switch to NB-IoT mode, uncomment the `MODE_NB_IOT` macro definition in the `TinyGsmClientSIM7028.h` file. If debugging is needed, you can uncomment the `DUMP_AT_COMMANDS` macro definition in the `TinyGsmClientSIM7028.h` file.

```
1  #include "ATOM_DTU_NB.h"
2  #include <TinyGsmClient.h>
3  #include <M5AtomS3.h>
4  #include <sys/time.h>
5  #include <time.h>
6  #include <ArduinoHttpClient.h>
7
8  // #define MODE_NB_IOT      // By default, CAT mode is used. If using NB-IOT
9  // mode, open this macro definition or TinyGsmClientSIM7028.h line 32 // #define
10 // MODE_NB_IOT
11 // #define DUMP_AT_COMMANDS // If you need to debug, you can open this macro
12 // definition and TinyGsmClientSIM7028.h line 13
13 // // #define TINY_GSM_DEBUG Serial
14 #ifdef DUMP_AT_COMMANDS
15 #include <StreamDebugger.h>
16 StreamDebugger debugger(SerialAT, SerialMon);
17 TinyGsm modem(debugger, ATOM_DTU_SIM7028_RESET);
18 #else
19 TinyGsm modem(SerialAT, ATOM_DTU_SIM7028_RESET);
20 #endif
21 // Server details
22 const char server[] = "api.m5stack.com";
23 const char resource[] = "/v1";
24 const int port = 80;
25 TinyGsmClient client(modem);
26 HttpClient http(client, server, port);
27
28 void modemConnect(void);
29
30 // Your GPRS credentials, if any
31 const char apn[] = "YourAPN";
32 const char gprsUser[] = "";
33 const char gprsPass[] = "";
34
35 struct tm now;
36 char s_time[50];
37
38 void log(String info) { SerialMon.println(info); }
39
40 void setup() {
41     AtomS3.begin(true); // Init M5AtomS3Lite.
42     AtomS3.dis.setBrightness(100);
43     AtomS3.dis.drawpix(0x0000ff);
44     Serial.println(">>ATOM DTU NB MQTT TEST");
45     SerialAT.begin(SIM7028_BAUDRATE, SERIAL_8N1, ATOM_DTU_SIM7028_RX,
46                   ATOM_DTU_SIM7028_TX);
47
48     modemConnect();
49 }
50
```

```

51 void loop() {
52     AtomS3.update();
53     SerialMon.print(F("Performing HTTP GET request... "));
54     int err = http.get(resource);
55     if (err != 0) {
56         SerialMon.println(F("failed to connect"));
57         delay(10000);
58         return;
59     }
60
61     int status = http.responseStatusCode();
62     SerialMon.print(F("Response status code: "));
63     SerialMon.println(status);
64     if (!status) {
65         delay(10000);
66         return;
67     }
68
69     SerialMon.println(F("Response Headers:"));
70     while (http.headerAvailable()) {
71         String headerName = http.readHeaderName();
72         String headerValue = http.readHeaderValue();
73         SerialMon.println("    " + headerName + " : " + headerValue);
74     }
75
76     int length = http.contentLength();
77     if (length >= 0) {
78         SerialMon.print(F("Content length is: "));
79         SerialMon.println(length);
80     }
81     if (http.isResponseChunked()) {
82         SerialMon.println(F("The response is chunked"));
83     }
84
85     String body = http.responseBody();
86     SerialMon.println(F("Response:"));
87     SerialMon.println(body);
88
89     SerialMon.print(F("Body length is: "));
90     SerialMon.println(body.length());
91
92     // Shutdown
93
94     http.stop();
95     SerialMon.println(F("Server disconnected"));
96 }
97
98
99 void modemConnect(void) {
100     // Get card number
101     String ccid = modem.getSimCCID();
102     Serial.println("CCID: " + ccid);
103     // Acquire signal strength

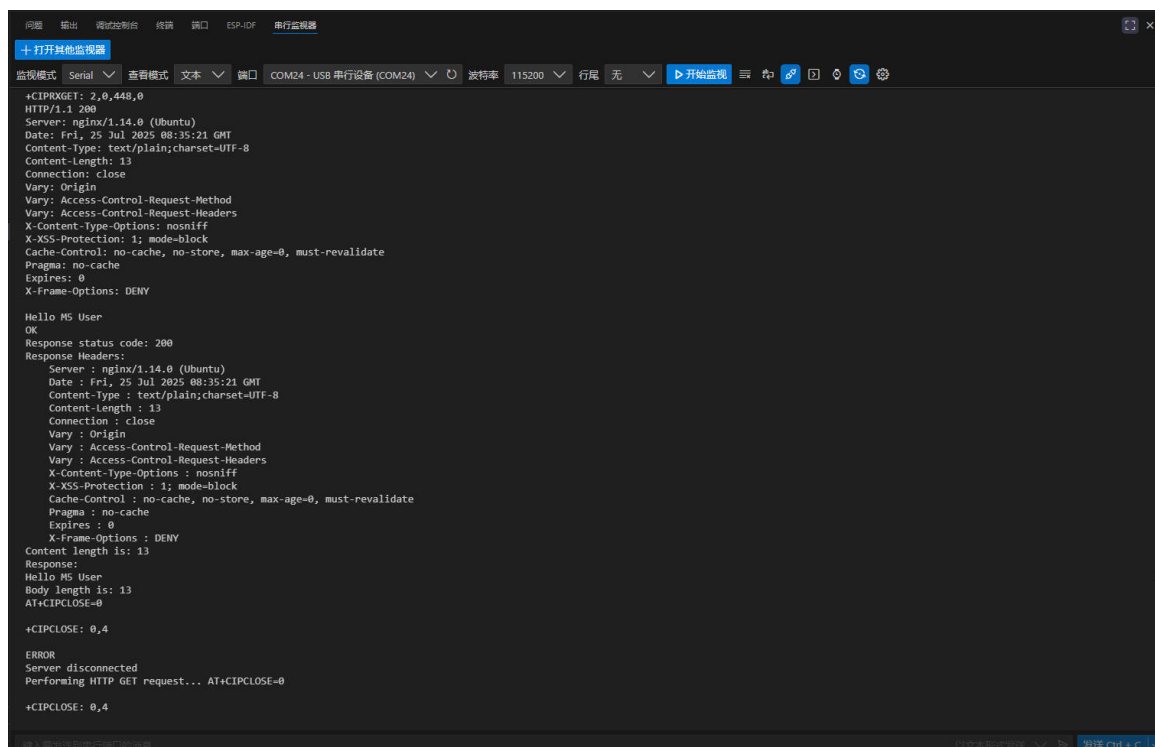
```

```

104     int csq = modem.getSignalQuality();
105     Serial.println("Signal quality: " + String(csq));
106     unsigned long start = millis();
107     log("Initializing modem...");
108     while (!modem.init()) {
109         log("waiting..." + String((millis() - start) / 1000) + "s");
110     };
111
112     start = millis();
113     log("Waiting for network...");
114     while (!modem.waitForNetwork()) {
115         log("waiting..." + String((millis() - start) / 1000) + "s");
116     }
117     log("success");
118 #ifdef MODE_NB_IOT
119 #else
120     SerialMon.println("Waiting for GPRS connect...");
121     if (!modem.gprsConnect(apn, gprsUser, gprsPass)) {
122         SerialMon.println("waiting..." + String((millis() - start) / 1000) + "s");
123     }
124     SerialMon.println("success");
125 #endif
126
127 // Example Query the IP address of a device
128 String ip = modem.getLocalIP();
129
130 log("Device IP address: " + ip);
131
132 log("success");
133 }
134

```

When the HTTP request is successful, it will return a series of information from the server and output it via serial port.



The screenshot shows a serial monitor window with the following content:

```

+CIIPXGET: 2,0,448,0
HTTP/1.1 200
Server: nginx/1.14.0 (Ubuntu)
Date: Fri, 25 Jul 2025 08:35:21 GMT
Content-Type: text/plain; charset=UTF-8
Content-Length: 13
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

Hello MS User
OK
Response status code: 200
Response Headers:
  Server : nginx/1.14.0 (Ubuntu)
  Date : Fri, 25 Jul 2025 08:35:21 GMT
  Content-Type : text/plain; charset=UTF-8
  Content-Length : 13
  Connection : close
  Vary : Origin
  Vary : Access-Control-Request-Method
  Vary : Access-Control-Request-Headers
  X-Content-Type-Options : nosniff
  X-XSS-Protection : 1; mode=block
  Cache-Control : no-cache, no-store, max-age=0, must-revalidate
  Pragma : no-cache
  Expires : 0
  X-Frame-Options : DENY
Content length is: 13
Response:
Hello MS User
Body length is: 13
AT+CIPCLOSE=0
+CIPCLOSE: 0,4

ERROR
Server disconnected
Performing HTTP GET request... AT+CIPCLOSE=0
+CIPCLOSE: 0,4

```

| 4. MQTT Service

This module also supports MQTT protocol communication. The examples include support for three MQTT service platforms: Baidu Cloud, OneNET and ThingsCloud. This tutorial is based on Baidu Cloud IoT Platform. If you need to use other platforms, please refer to other examples in the `example` folder.

The SIM card and debug mode selection are the same as for HTTP.

Replace `MQTT_BROKER`, `mqtt_devId`, `mqtt_pubId`, and `mqtt_password` with your personal MQTT account information. For any questions, please refer to [this tutorial](#).

```
1  #include <M5AtomS3.h>
2  #include "ATOM_DTU_NB.h"
3  #include <PubSubClient.h>
4  #include <TinyGsmClient.h>
5  #include <time.h>
6  #include <sys/time.h>
7  #define MQTT_BROKER    "*****"    //Baidu Cloud address
8  #define MQTT_PORT      1883        //Port number
9
10 #define UPLOAD_INTERVAL  2000
11 #define mqtt_devid        "*****"    //Device ID
12 #define mqtt_pubid        "*****"    //username
13 #define mqtt_password     "*****"    //password
14 int postMsgId = 0;    //Keep track of how many posts have been made
15 // This is the template used by post to upload data
16 #define ONENET_POST_BODY_FORMAT "{ \"id\":%d, \"dp\":%s}"
17 // Receiving and sending properties set the subject
18 // Receive device properties to get the command topic
19 #define ONENET_TOPIC_GET "$iot/" mqtt_devid "/msg"
20 // Send data subject on the device
21 #define ONENET_TOPIC_POST "$iot/" mqtt_devid "/events"
22 int num=0;
23 uint32_t lastReconnectAttempt = 0;
24
25 #define DUMP_AT_COMMANDS    //If you need to debug, you can open this macro definition and TinyGsmC
26 #ifdef DUMP_AT_COMMANDS
27 #include <StreamDebugger.h>
28     StreamDebugger debugger(SerialAT, SerialMon);
29     TinyGsm modem(debugger, ATOM_DTU_SIM7028_RESET);
30 #else
31     TinyGsm modem(SerialAT, ATOM_DTU_SIM7028_RESET);
32 #endif
33
34 TinyGsmClient tcpClient(modem);
35 PubSubClient mqttClient(MQTT_BROKER, MQTT_PORT, tcpClient);
36
37 void mqttCallback(char *topic, byte *payload, unsigned int len);
38 bool mqttConnect(void);
39 void nbConnect(void);
40
41 void log(String info) {
42     SerialMon.println(info);
43 }
44
45 void setup() {
46     AtomS3.begin(true);
47     Serial.println(">>ATOM DTU NB MQTT TEST");
48     SerialAT.begin(SIM7028_BAUDRATE, SERIAL_8N1, ATOM_DTU_SIM7028_RX,
49                     ATOM_DTU_SIM7028_TX);
50     AtomS3.display.drawpix(0x0000ff);
```

```

51     nbConnect();
52     mqttClient.setServer(MQTT_BROKER, MQTT_PORT);
53     mqttClient.setCallback(mqttCallback);
54 }
55
56 void loop() {
57     static unsigned long timer = 0;
58
59     if (!mqttClient.connected()) {
60         log(">>MQTT NOT CONNECTED");
61         log(mqttClient.state());
62         AtomS3.dis.drawpix(0xff0000);
63         uint32_t t = millis();
64         if (t - lastReconnectAttempt > 10000L) {
65             lastReconnectAttempt = t;
66             if (mqttConnect()) {
67                 lastReconnectAttempt = 0;
68             }
69         }
70         delay(100);
71     }
72     if (millis() >= timer) {
73         timer = millis() + UPLOAD_INTERVAL;
74         if (mqttClient.connected())
75         {
76             // First concatenate the json string
77             char param[120];
78             char jsonBuf[178];
79             sprintf(param, "{\"num\":[{\"v\":%d}]}",num); // We write the data to be uploaded in the
80
81             postMsgId += 1;
82             num+=1;
83             if(num>256){
84                 num=0;
85             }
86             sprintf(jsonBuf, ONENET_POST_BODY_FORMAT, postMsgId, param);
87
88             log("public the data:");
89             log(jsonBuf);
90             log("\n");
91
92             mqttClient.publish(ONENET_TOPIC_POST, jsonBuf);
93             //Send data to the topic
94             delay(100);
95
96         }
97     }
98     AtomS3.dis.drawpix(0x00ff00);
99     mqttClient.loop();
100 }
101
102 void mqttCallback(char *topic, byte *payload, unsigned int len) {
103     char info[len + 1];

```

```

104     memcpy(info, payload, len);
105     info[len] = '\0';
106     log("Message arrived:"+String(info));
107     log("Topic received: " + String(topic));
108 }
109
110 bool mqttConnect(void) {
111     log("Connecting to ");
112     log(MQTT_BROKER);
113     bool status =mqttClient.connect(mqtt_devid, mqtt_pubid, mqtt_password);
114     if (status == false) {
115         int errorCode = mqttClient.state();
116         log("MQTT Connection failed with error code: " + String(errorCode));
117         return false;
118     }
119     log("MQTT CONNECTED!");
120     mqttClient.subscribe(ONENET_TOPIC_GET);
121     return mqttClient.connected();
122 }
123
124 void nbConnect(void) {
125     unsigned long start = millis();
126     log("Initializing modem...");
127     while (!modem.init()) {
128         log("waiting...." + String((millis() - start) / 1000) + "s");
129     };
130
131     start = millis();
132     log("Waiting for network...");
133     while (!modem.waitForNetwork()) {
134         log("waiting...." + String((millis() - start) / 1000) + "s");
135     }
136     log("success");
137     String ccid = modem.getSimCCID();
138     Serial.println("CCID: " + ccid);
139     int csq = modem.getSignalQuality();
140     Serial.println("Signal quality: " + String(csq));
141 }
142

```

When the MQTT service is successfully connected, you can use MQTT Explorer to add subscription topics and view data. (For any questions, please refer to [this interface](#))

☰

MQTT Explorer

🔍

Search...

⏸

▼

► \$sys (1 topic, 4 messages)

▼ Siot

▼ MQTT

events = {"id":33,"dp":{"num":[{"v":32}]}}