DIY ESP-Smartwatch

This ESPWatch starter Kit is for the beginners to learn how to create an ESP watch with detailed guide, it is for the starters to dive into the electronic world, with 1~2 hours' learning, a none-electronic starter can create this ESP watch, to enjoy the joy of creating something by oneself.

The ESPWatch is based on ESP12 WIFI module, it gets the real time from Internet server, and can also remote control the local instruments, such as Relay/LED/Fan. With this learning, you will learn the basic skill of components soldering/Arduino Programming/ WIFI usage/Basic http protocol, the first& easy step to get into the electronic/programming world.

There we also prepared simple case/watchband, so that you can get it works on your wrist within few minutes. Are you ready to show this cool watch produced by yourself?

Spec:

1. Kits with detailed guide for learners;
2. Video for learning;
3. Based on Arduino IDE/ESP;
4. Real time watch + Remote control;
5. Open hardware+ Open software;
6. For ages 12+;

Pack list:

0.96inch IIC OLED X1

Button x3

Micro USB x1

watchbandx1

Some resistors and capacitors

ESP-12S x1

3.7V Lipo Battery x1

Switch x1

Acrylic shell x1

Some of Copper column and screw

DIY ESP-Smartwatch Guide

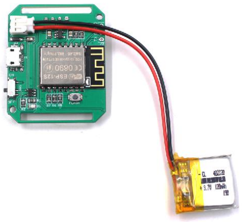
1. Set up Arduino IDE

Follow the guide to setup Arduino IDE for ESP8266

https://arduino-esp8266.readthedocs.io/en/latest/installing.html

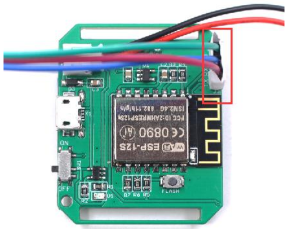
2. Hardware connection

2.1 Connect a battery



2.2 Connect GND, RX and TX to a USB-to-Serial adapter

Watch	USB to Serial
GND →	GND
TX →	RX
RX →	TX



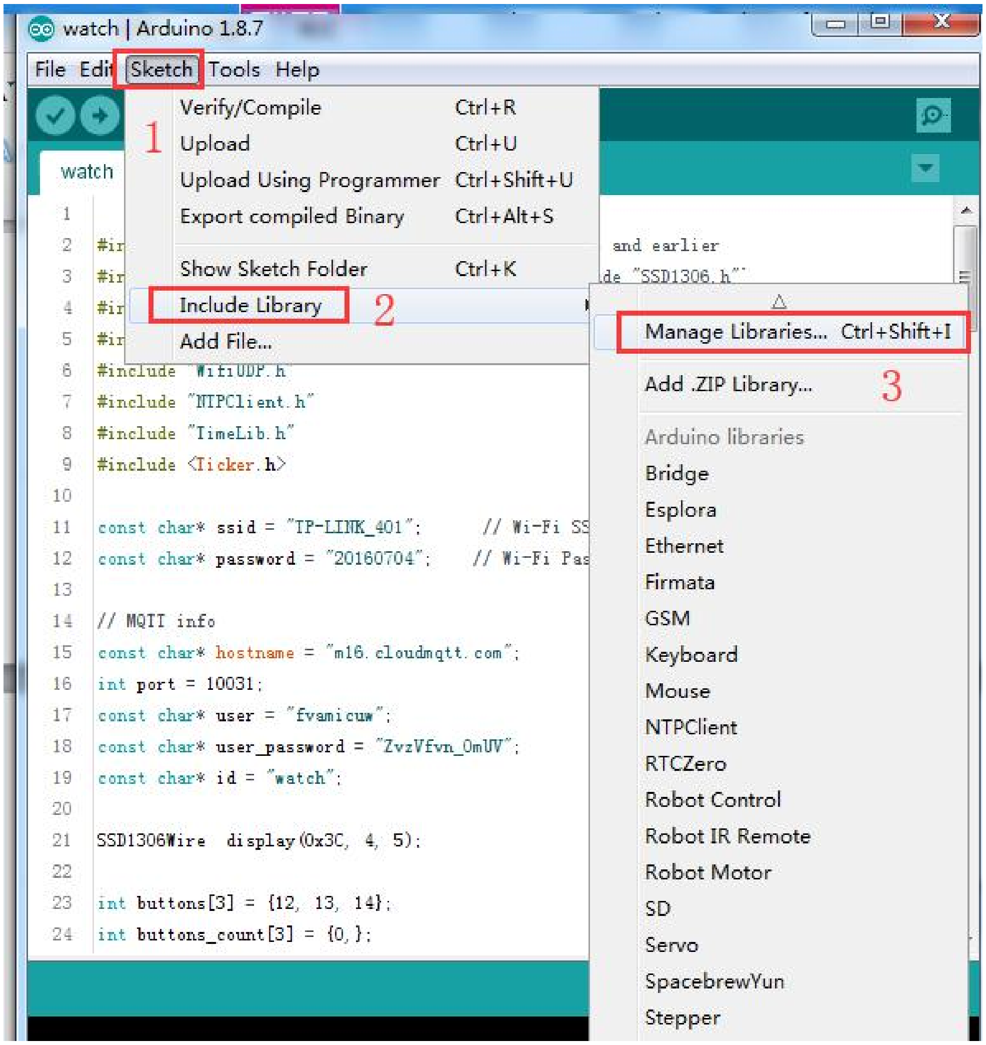
3. Download the code to smart watch and relay

This example will get time from a NTP server, display date and time on OLED, and control the relay through MQTT message bus.

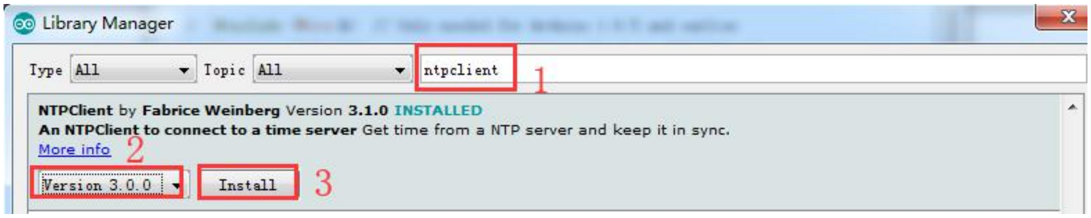
3.1 Dependencies

- arduino-mqtt
- ThingPulse ESP8266 OLED SSD1306
- TimeLib

We can search "ntpclient","lwmqtt", "esp ssd1306" and "timekeeping" to find these libraries in Library Manager. Then install them.



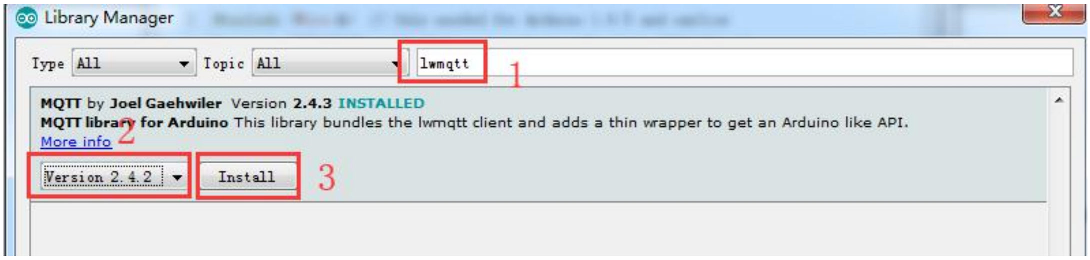
```

#ir                                              and earlier
#ir                                           de "SSD1306.h"
#ir
#ir
#include "WifiUDP.h"
#include "NTPClient.h"
#include "TimeLib.h"
#include <Ticker.h>

const char* ssid = "TP-LINK_401";       // Wi-Fi SS
const char* password = "20160704";     // Wi-Fi Pas

// MQTT info
const char* hostname = "m16.cloudmqtt.com";
int port = 10031;
const char* user = "fvamicuw";
const char* user_password = "ZvzVfvn_OmUV";
const char* id = "watch";

SSD1306Wire  display(0x3C, 4, 5);

int buttons[3] = {12, 13, 14};
int buttons_count[3] = {0,};
```



Type All Topic All ntpclient

NTPClient by **Fabrice Weinberg** Version **3.1.0** INSTALLED
An NTPClient to connect to a time server Get time from a NTP server and keep it in sync.
More info

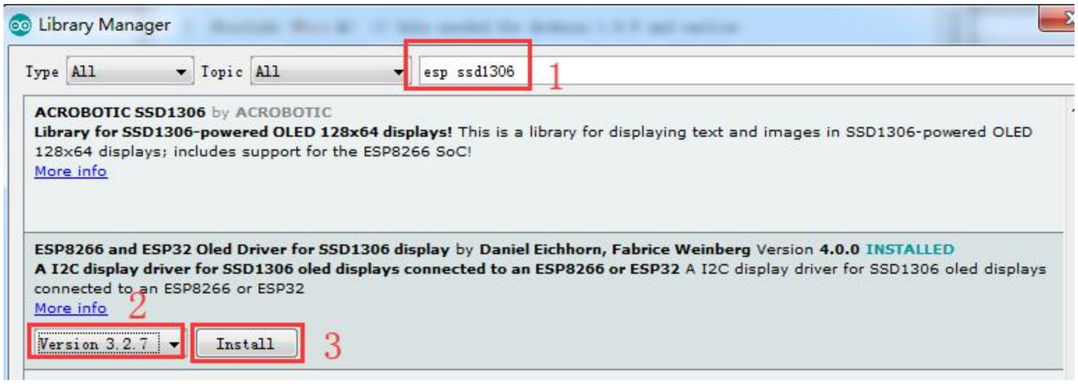
Version 3.0.0



Type All Topic All lwmqtt

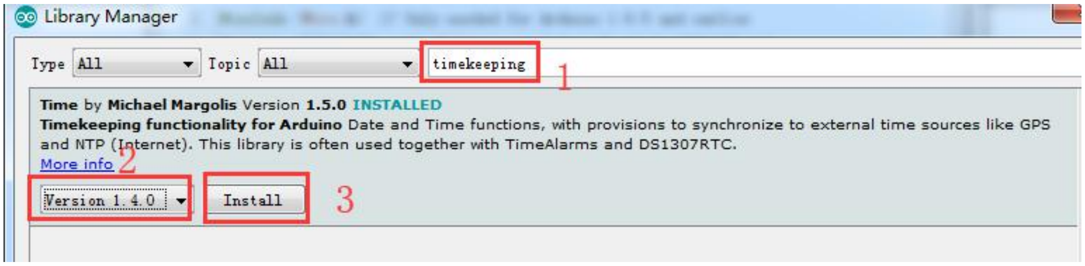
MQTT by **Joel Gaehwiler** Version **2.4.3** INSTALLED
MQTT library for Arduino This library bundles the lwmqtt client and adds a thin wrapper to get an Arduino like API.
More info

Version 2.4.2



ACROBOTIC SSD1306 by ACROBOTIC
Library for SSD1306-powered OLED 128x64 displays! This is a library for displaying text and images in SSD1306-powered OLED 128x64 displays; includes support for the ESP8266 SoC!
More info

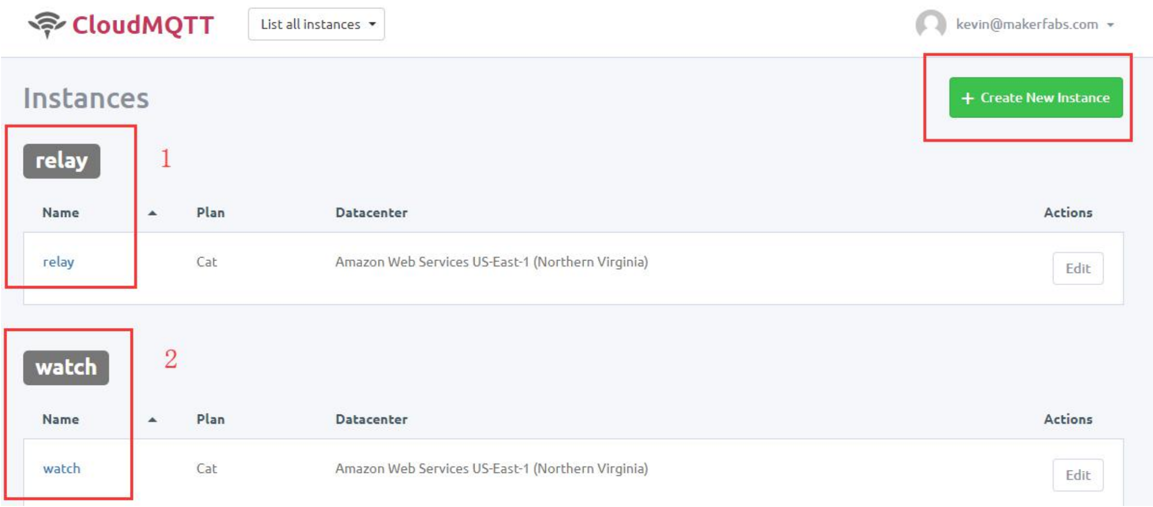
ESP8266 and ESP32 Oled Driver for SSD1306 display by **Daniel Eichhorn, Fabrice Weinberg** Version **4.0.0** INSTALLED
A I2C display driver for SSD1306 oled displays connected to an ESP8266 or ESP32 A I2C display driver for SSD1306 oled displays connected to an ESP8266 or ESP32
More info



Time by **Michael Margolis** Version **1.5.0** INSTALLED
Timekeeping functionality for Arduino Date and Time functions, with provisions to synchronize to external time sources like GPS and NTP (Internet). This library is often used together with TimeAlarms and DS1307RTC.
More info

3.2 Use CloudMQTT as the broker here.

3.2.1 Signup CloudMQTT and create the instance



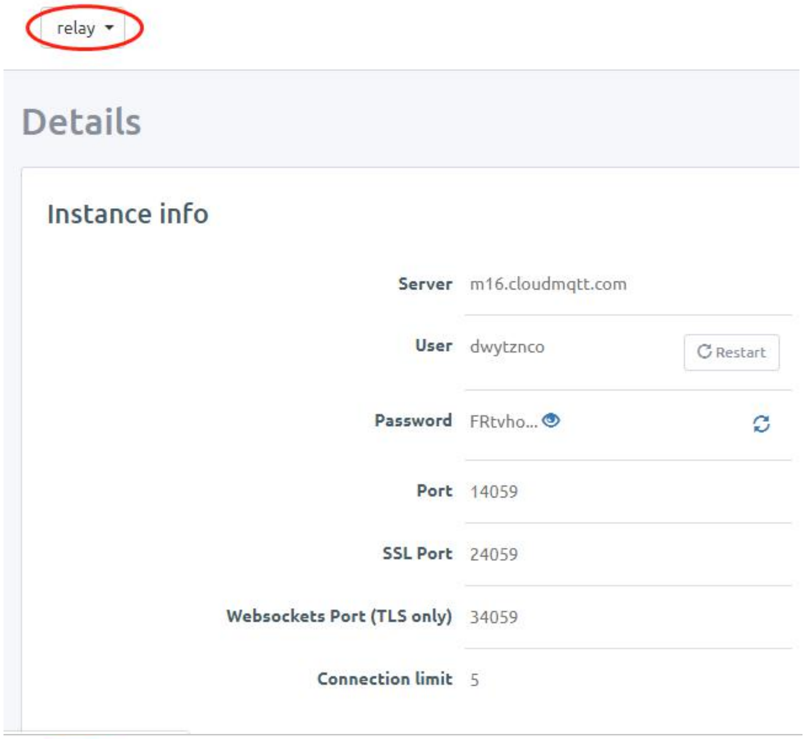
Instances

relay

Name	Plan	Datacenter	Actions
relay	Cat	Amazon Web Services US-East-1 (Northern Virginia)	Edit

watch

Name	Plan	Datacenter	Actions
watch	Cat	Amazon Web Services US-East-1 (Northern Virginia)	Edit



Details

Instance info

Server m16.cloudmqtt.com

User dwytznco

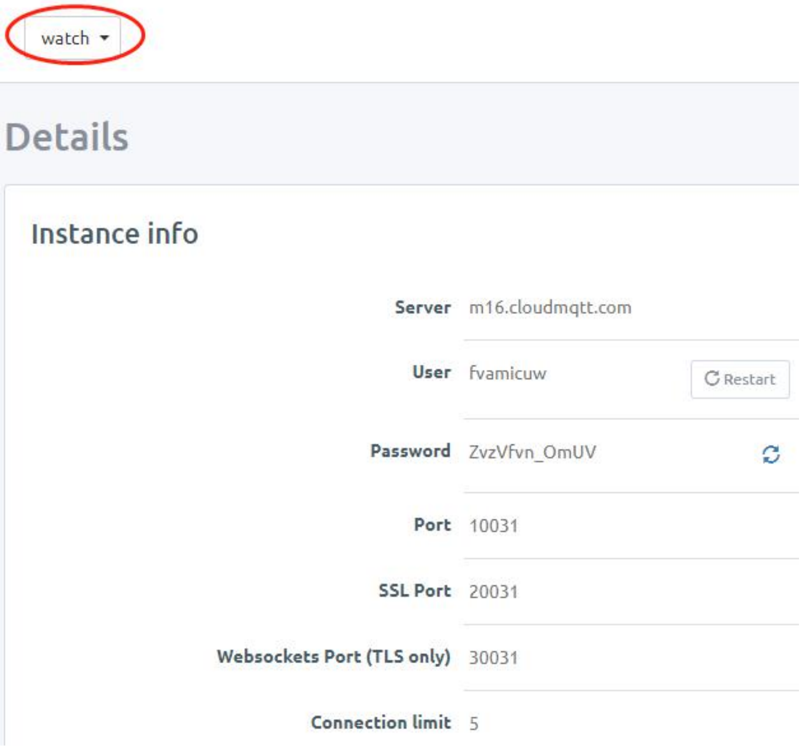
Password FRtvho...

Port 14059

SSL Port 24059

Websockets Port (TLS only) 34059

Connection limit 5



Details

Instance info

Server m16.cloudmqtt.com

User fvamicuw

Password ZvzVfvn_OmUV

Port 10031

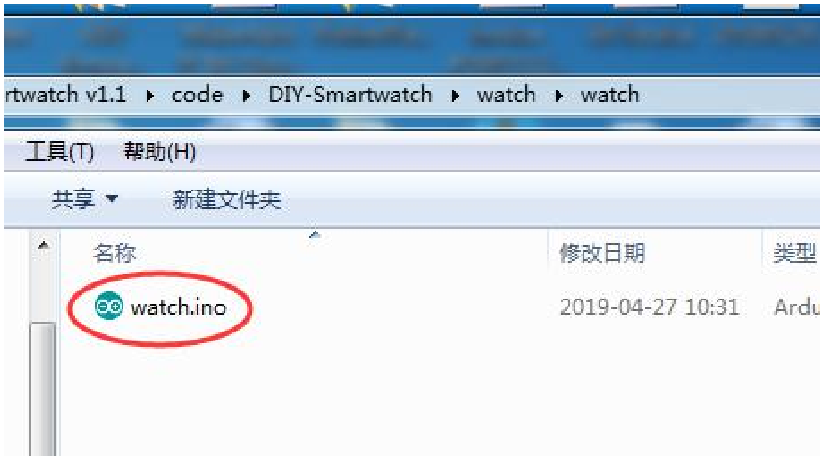
SSL Port 20031

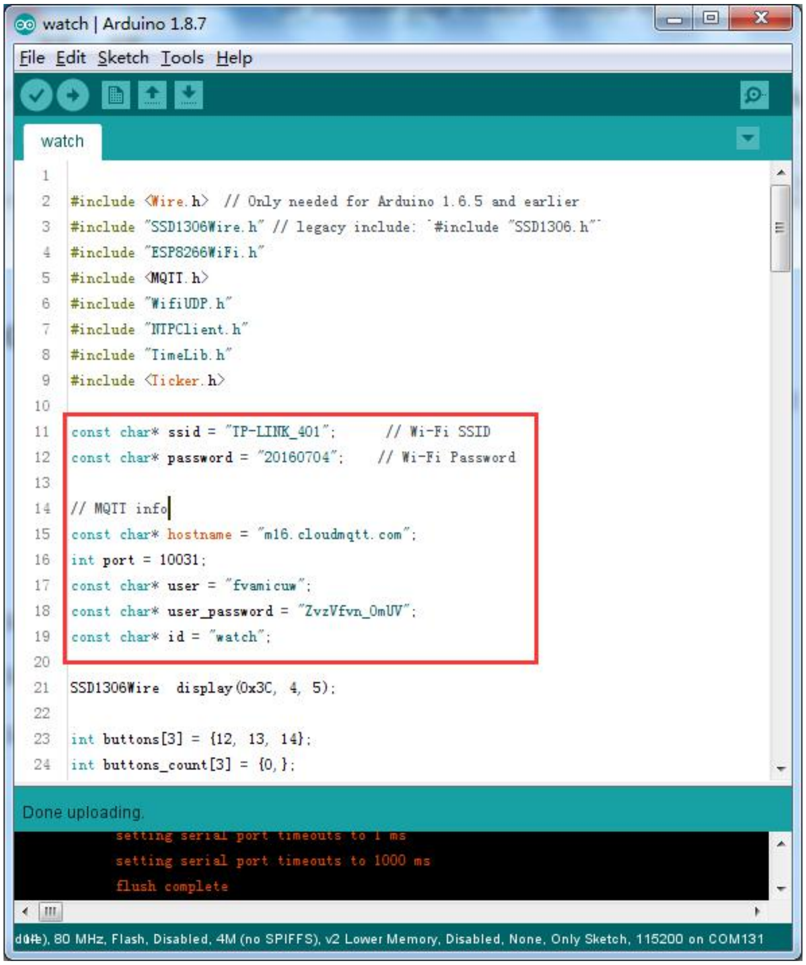
Websockets Port (TLS only) 30031

Connection limit 5

3.3 Download the sketches watch/watch.ino to the watch

3.3.1 Open the watch.ino sketches, change `SSID` and `passowrd` of the Wi-Fi, and modify `hostname`, `port`, `user` and `user_password` for MQTT





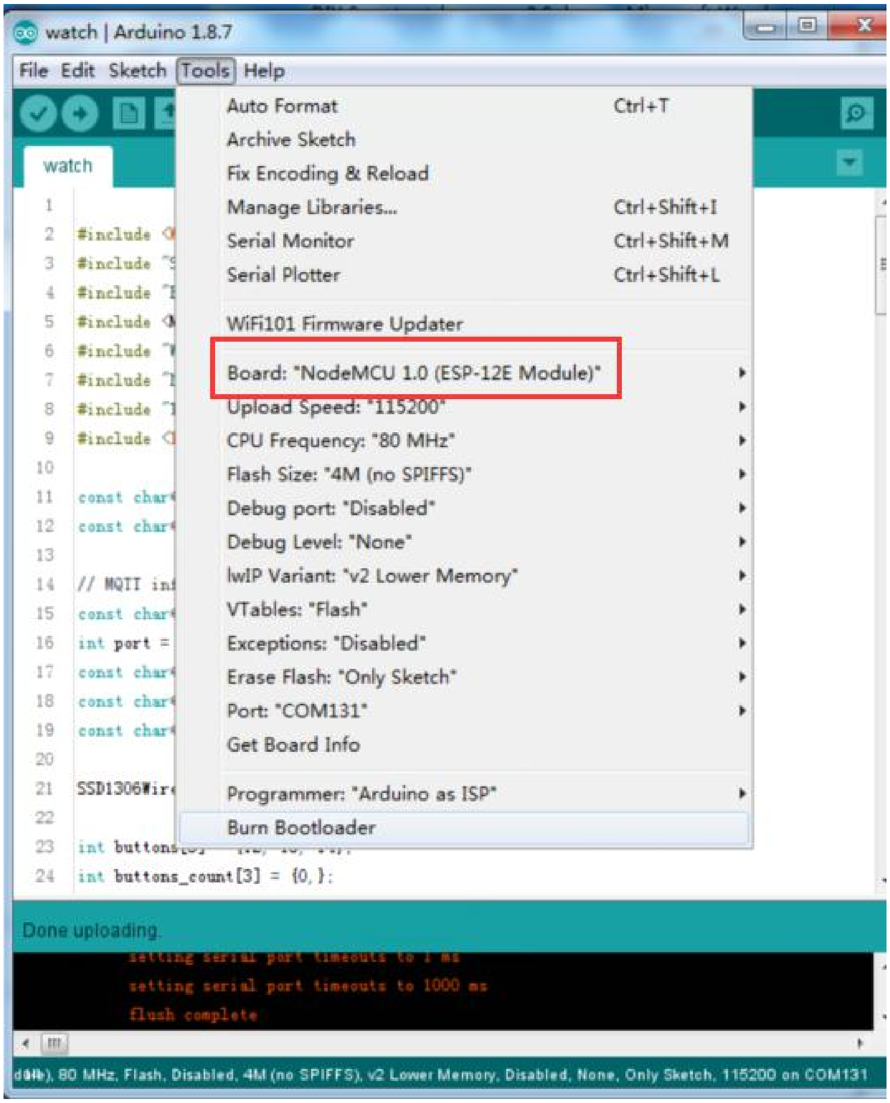
```

#include <Wire.h>  // Only needed for Arduino 1.6.5 and earlier
#include "SSD1306Wire.h" // legacy include: `#include "SSD1306.h"`
#include "ESP8266WiFi.h"
#include <MQTT.h>
#include "WifiUDP.h"
#include "NTPClient.h"
#include "TimeLib.h"
#include <Ticker.h>

const char* ssid = "TP-LINK_401";       // Wi-Fi SSID
const char* password = "20160704";     // Wi-Fi Password

// MQTT info
const char* hostname = "m16.cloudmqtt.com";
int port = 10031;
const char* user = "fvamicuw";
const char* user_password = "ZvzVfvn_OmUV";
const char* id = "watch";

SSD1306Wire  display(0x3C, 4, 5);

int buttons[3] = {12, 13, 14};
int buttons_count[3] = {0,};
```

Done uploading.

```
        setting serial port timeouts to 1 ms
        setting serial port timeouts to 1000 ms
        flush complete
```

d0#e), 80 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM131

3.3.2 Select the right board and com port



Board: "NodeMCU 1.0 (ESP-12E Module)"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4M (no SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Disabled"
Erase Flash: "Only Sketch"
Port: "COM131"
Programmer: "Arduino as ISP"

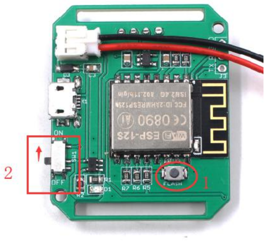
```

#include <
#include "
#include "
#include <
#include "
#include "
#include "
#include <

const char
const char

// MQTT inf
const char
int port =
const char
const char
const char

SSD1306Wir

int buttons
int buttons_count[3] = {0, };
```

Done uploading.

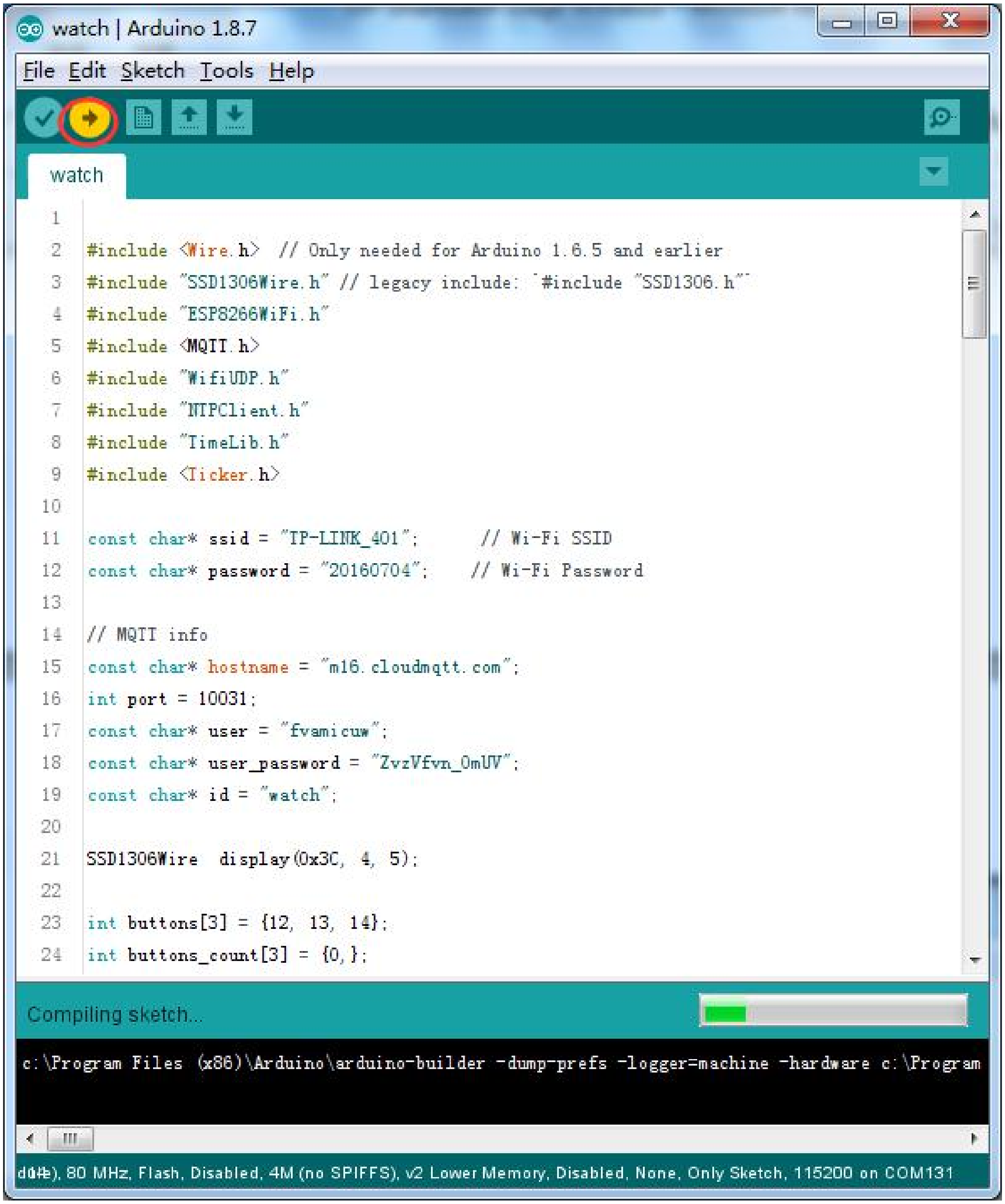
```
setting serial port timeouts to 1 ms
setting serial port timeouts to 1000 ms
flush complete
```

d04b), 80 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM131

3.3.4 Hold the FLASH button; Power on the ESP8266 by the SW1 to put ESP8226 into bootloader mode



3.3.5 Click the upload button, upload the code



```

#include <Wire.h>  // Only needed for Arduino 1.6.5 and earlier
#include "SSD1306Wire.h" // legacy include: `#include "SSD1306.h"`
#include "ESP8266WiFi.h"
#include <MQTT.h>
#include "WifiUDP.h"
#include "NTPClient.h"
#include "TimeLib.h"
#include <Ticker.h>

const char* ssid = "TP-LINK_401";       // Wi-Fi SSID
const char* password = "20160704";     // Wi-Fi Password

// MQTT info
const char* hostname = "m16.cloudmqtt.com";
int port = 10031;
const char* user = "fvamicuw";
const char* user_password = "ZvzVfvn_OmUV";
const char* id = "watch";

SSD1306Wire  display(0x3C, 4, 5);

int buttons[3] = {12, 13, 14};
int buttons_count[3] = {0,};
```

Compiling sketch...

```
c:\Program Files (x86)\Arduino\arduino-builder -dump-prefs -logger=machine -hardware c:\Program
```

d04), 80 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM131

3.4 Download the sketches [relay/relay.ino] to the relay module

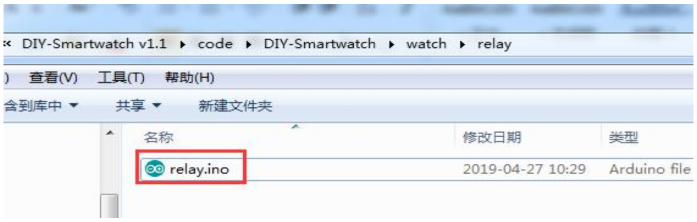
3.4.1 Open the sketches, change `SSID` and `passowrd` of the Wi-Fi, and modify `hostname`, `port`, `user` and `user_password` for MQTT

Noted: port must same as the watch.

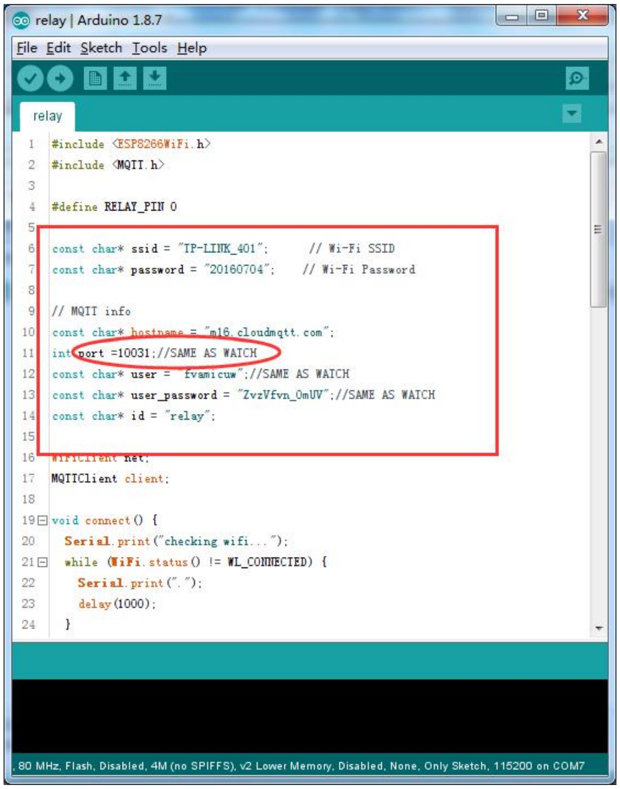
3.4.2 ESP-01S: Use the ESP8266 debugger, auto reset when upload the code. Same as the NodeMCU.

3.4.3 Plug the ESP-01 the ESP8266 debugger.

3.4.4 Open the sketches relay.ino


« DIY-Smartwatch v1.1 ▸ code ▸ DIY-Smartwatch ▸ watch ▸ relay
) 查看(V) 工具(T) 帮助(H)
含到库中 ▾ 共享 ▾ 新建文件夹

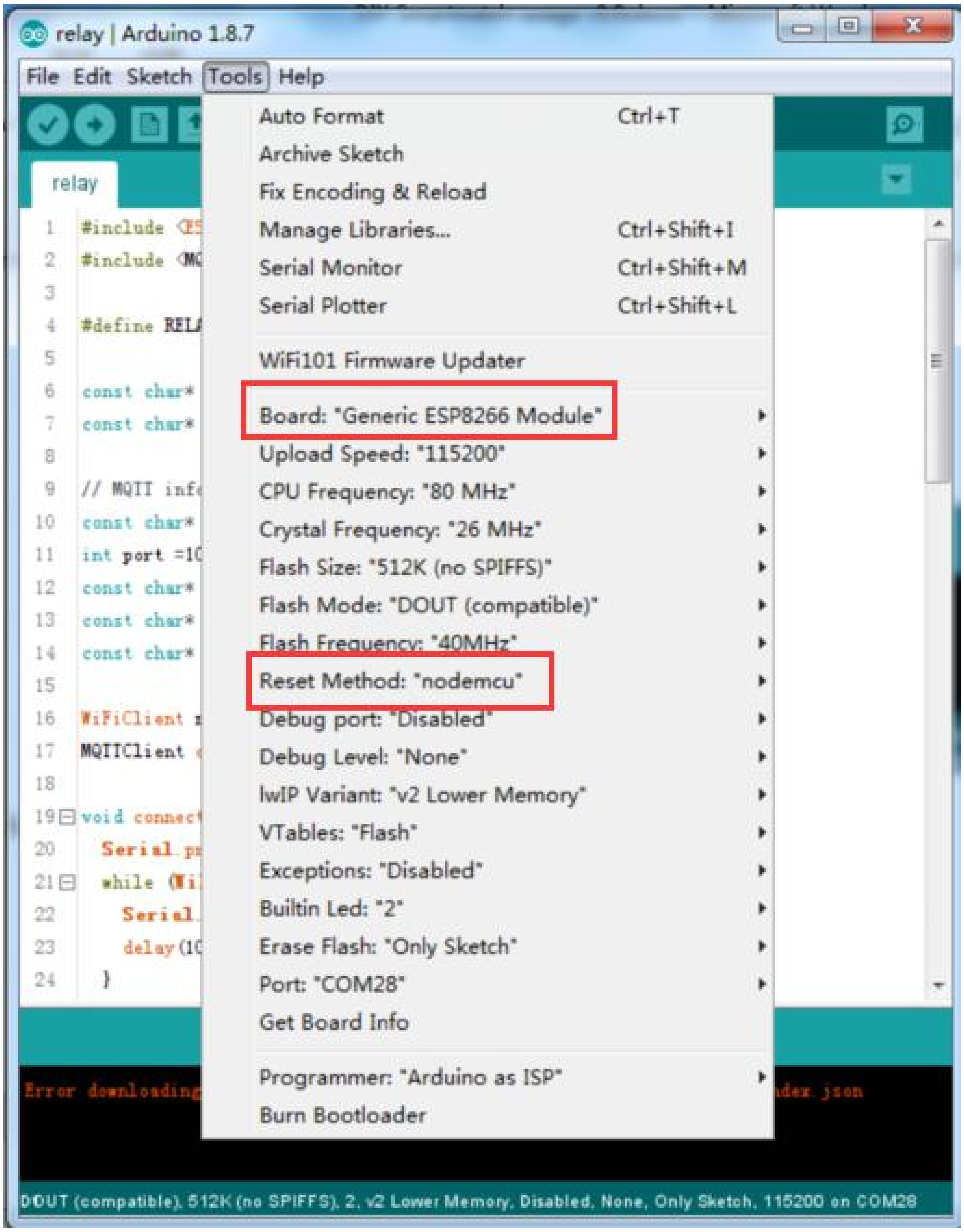
名称	修改日期	类型
relay.ino	2019-04-27 10:29	Arduino file



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

#define RELAY_PIN 0

const char* ssid = "TP-LINK_401";       // Wi-Fi SSID
const char* password = "20160704";     // Wi-Fi Password

// MQTT info
const char* hostname = "m16.cloudmqtt.com";
int port =10031;//SAME AS WATCH
const char* user = "fvamicuw";//SAME AS WATCH
const char* user_password = "ZvzVfvn_OmUV";//SAME AS WATCH
const char* id = "relay";

WiFiClient net;
MQTTClient client;

void connect() {
  Serial.print("checking wifi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }
```

, 80 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM7

3.4.5 Select the right board

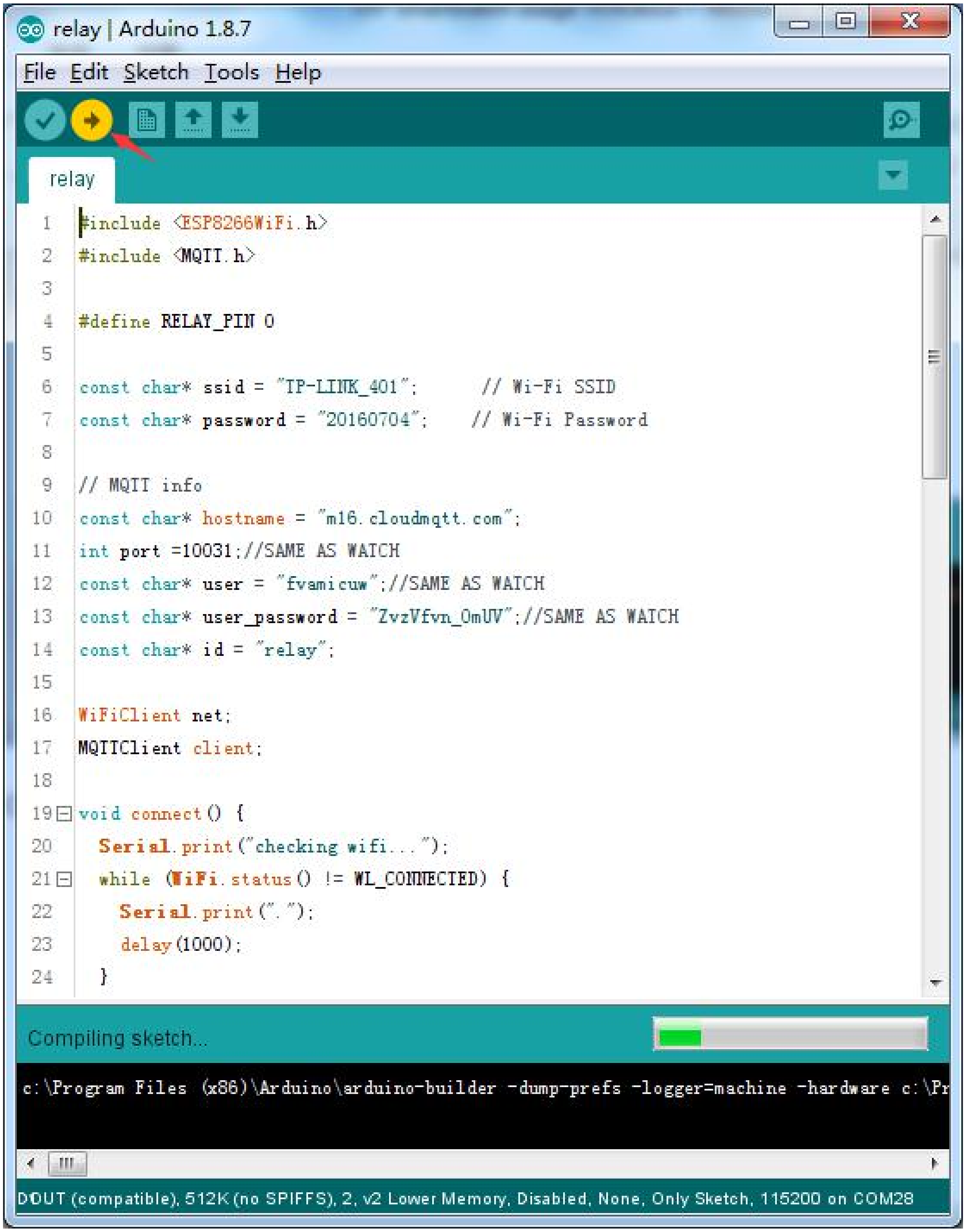


Board: "Generic ESP8266 Module"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Crystal Frequency: "26 MHz"
Flash Size: "512K (no SPIFFS)"
Flash Mode: "DOUT (compatible)"
Flash Frequency: "40MHz"
Reset Method: "nodemcu"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Disabled"
Builtin Led: "2"
Erase Flash: "Only Sketch"
Port: "COM28"
Programmer: "Arduino as ISP"

```
Error downloading                    dex.json
```

DOUT (compatible), 512K (no SPIFFS), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM28

3.4.6 Click upload button upload the code.



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

#define RELAY_PIN 0

const char* ssid = "TP-LINK_401";      // Wi-Fi SSID
const char* password = "20160704";    // Wi-Fi Password

// MQTT info
const char* hostname = "m16.cloudmqtt.com";
int port =10031;//SAME AS WATCH
const char* user = "fvamicuw";//SAME AS WATCH
const char* user_password = "ZvzVfvn_OmUV";//SAME AS WATCH
const char* id = "relay";

WiFiClient net;
MQTTClient client;

void connect() {
  Serial.print("checking wifi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }
```

Compiling sketch...

```
c:\Program Files (x86)\Arduino\arduino-builder -dump-prefs -logger=machine -hardware c:\Pr
```

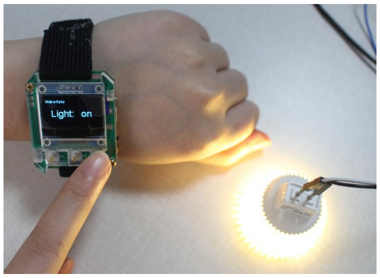
DOUT (compatible), 512K (no SPIFFS), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM28

4. Now you can use you smart watch to control the relay.

4.1 Smart watch time display:

4.2 Use the buttons "S1" and "S2" to control the Light and Fan.

1) Press the S1 select the Light control UI, press the S2 turn ON or turn OFF the light.



2) Press the S1 select the Fan control UI, press the S2 turn ON or turn OFF the Fan.