1. 通过Wi-Fi + AP + HTTP服务器，实现本地数据发布

//温湿度库对应的头文件

#include <dht11.h>

//网络服务所需头文件

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

//设置AP的SSID和密码

const char \*ssid = "DHT11";

const char \*password = "12345678";

//全局对象，用提供web服务

ESP8266WebServer server(80);

//全局对象，用于获取温湿度

dht11 DHT11;

//温湿度的数据针脚

#define DHT11PIN D4

//访问http://192.168.4.1会看到的内容

void handleRoot() {

float h = getHumidity();

float t = getTemperature();

String str = "<head><meta charset=\"utf-8\" /><title>ESP-12E试验</title></head><h1>当前湿度(%):";

str += String(h);

str += "<br>当前温度(℃):";

str += String(t);

str += "</h1>";

server.send(200, "text/html", str);

}

void setup() {

delay(1000);

Serial.begin(115200);

Serial.println();

Serial.print("Configuring access point...");

//启动AP热点

WiFi.softAP(ssid, password);

//显示热点信息

IPAddress myIP = WiFi.softAPIP();

Serial.print("AP IP address: ");

Serial.println(myIP);

//绑定HTTP服务器/目录下所发布的信息，并启动服务

server.on("/", handleRoot);

server.begin();

Serial.println("HTTP server started");

}

//获取湿度

float getHumidity(){

int chk = DHT11.read(DHT11PIN);

return (float)DHT11.humidity;

}

//获取温度

float getTemperature(){

int chk = DHT11.read(DHT11PIN);

return (float)DHT11.temperature;

}

void loop() {

server.handleClient();

}

## 二、通过Wi-Fi + AP + HTTP服务器，实现本地传感器控制

//网络服务所需头文件

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

//设置AP的SSID和密码

const char \*ssid = "NodeMCU";

const char \*password = "12345678";

//全局对象，用提供web服务

ESP8266WebServer server(80);

//全局字符串对象，用以完成网页所需的html脚本

String strHtml1 = "\

<!DOCTYPE html>\

<html>\

<head>\

<meta charset=\"utf-8\" />\

<title></title>\

</head>\

<body>\

<form id=form1>\

<input type=\"button\" value=\"开灯\" onclick=\'location.href=(\"/Digital?value=1\")\' />\

<input type=\"button\" value=\"关灯\" onclick=\'location.href=(\"/Digital?value=0\")\' />\

</form>\

<h1>D4 is ";

//全局字符串对象，用以完成网页所需的html脚本

String strHtml2 ="</h1>\

</body>\

</html>\

";

//全局变量，表示LED的亮灭

int valueLED = 0;

void setup() {

delay(1000);

Serial.begin(115200);

Serial.println();

Serial.print("Configuring access point...");

//启动AP热点

WiFi.softAP(ssid, password);

//显示热点信息

IPAddress myIP = WiFi.softAPIP();

Serial.print("AP IP address: ");

Serial.println(myIP);

//设置D4端口模式为输出

pinMode(D4, OUTPUT);

digitalWrite(D4, 0);

//绑定HTTP服务器/目录下所发布的信息

server.on("/", handleRoot);

//绑定HTTP服务器/Digital目录下所发布的信息

server.on("/Digital", HTTP\_GET, handleDigital);

//启动服务

server.begin();

Serial.println("HTTP Server started");

}

//用户跳转到http://192.168.4.1/Digital时页面内容

void handleDigital()

{

String str = server.arg("value");

valueLED = str.toInt();

digitalWrite(D4, valueLED);

String strHtml = strHtml1 + String(valueLED?"HIGH" : "LOW") + strHtml2;

server.send(200, "text/html", strHtml);

}

//用户跳转到http://192.168.4.1时页面内容

void handleRoot() {

String strHtml = strHtml1 + String(valueLED?"HIGH" : "LOW") + strHtml2;

server.send(200, "text/html", strHtml);

}

void loop() {

server.handleClient();

}

### 三、通过MQTT客户端完成与IOT的互联

#include <ESP8266WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WiFi Access Point \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define WLAN\_SSID "" //你的WI-FI的SSID

#define WLAN\_PASS "" //你的WI-FI的密码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Adafruit.io Setup \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define AIO\_SERVER "io.adafruit.com"

#define AIO\_SERVERPORT 1883 // use 8883 for SSL

#define AIO\_USERNAME "" //你在io.adafruit.com上注册的用户名

#define AIO\_KEY "" //你在io.adafruit.com所获得的AIO

/\*\*\*\*\*\*\*\*\*\*\*\* Global State (you don't need to change this!) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.

WiFiClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Feeds \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Setup a feed called 'photocell' for publishing.

// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>

Adafruit\_MQTT\_Publish photocell = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/test");

// Setup a feed called 'onoff' for subscribing to changes.

Adafruit\_MQTT\_Subscribe onoffbutton = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Switch");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sketch Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration

// for some reason (only affects ESP8266, likely an arduino-builder bug).

void MQTT\_connect();

void setup() {

Serial.begin(115200);

delay(10);

Serial.println(F("Adafruit MQTT demo"));

// Connect to WiFi access point.

Serial.println(); Serial.println();

Serial.print("Connecting to ");

Serial.println(WLAN\_SSID);

WiFi.begin(WLAN\_SSID, WLAN\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.println("WiFi connected");

Serial.println("IP address: "); Serial.println(WiFi.localIP());

// Setup MQTT subscription for onoff feed.

mqtt.subscribe(&onoffbutton);

pinMode(D6, OUTPUT);

}

uint32\_t x=0;

void loop() {

// Ensure the connection to the MQTT server is alive (this will make the first

// connection and automatically reconnect when disconnected). See the MQTT\_connect

// function definition further below.

MQTT\_connect();

// this is our 'wait for incoming subscription packets' busy subloop

// try to spend your time here

Adafruit\_MQTT\_Subscribe \*subscription;

while ((subscription = mqtt.readSubscription(5000))) {

if (subscription == &onoffbutton) {

Serial.print(F("Got: "));

Serial.println((char \*)onoffbutton.lastread);

String value = (char \*)onoffbutton.lastread;

Serial.println(value);

if(!value.compareTo("开"))

{

digitalWrite(D6,HIGH);

}

if(!value.compareTo("关"))

{

digitalWrite(D6,LOW);

}

}

}

// Now we can publish stuff!

int value = analogRead(A0);

x = map(value,100,1000,1,100);

Serial.print(F("\nSending photocell val "));

Serial.print(x);

Serial.print("...");

if (! photocell.publish(x++)) {

Serial.println(F("Failed"));

} else {

Serial.println(F("OK!"));

}

// ping the server to keep the mqtt connection alive

// NOT required if you are publishing once every KEEPALIVE seconds

/\*

if(! mqtt.ping()) {

mqtt.disconnect();

}

\*/

}

// Function to connect and reconnect as necessary to the MQTT server.

// Should be called in the loop function and it will take care if connecting.

void MQTT\_connect() {

int8\_t ret;

// Stop if already connected.

if (mqtt.connected()) {

return;

}

Serial.print("Connecting to MQTT... ");

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected

Serial.println(mqtt.connectErrorString(ret));

Serial.println("Retrying MQTT connection in 5 seconds...");

mqtt.disconnect();

delay(5000); // wait 5 seconds

retries--;

if (retries == 0) {

// basically die and wait for WDT to reset me

while (1);

}

}

Serial.println("MQTT Connected!");

}