多功能无线物联实验平台

合肥市高新区梦园小学 五（5）班　周米笑

平台简介

多功能无线物联实验平台是一套基于 ESP8266 WIFI 硬件模块，用 Arduino 开发的物联实验平台。该实验平台使得物联实验变得更简单，主要特点如下：

1. 通过 Wifi 与电脑端的服务程序进行无线连接。

无论是调试还是实际应用都可以通过 Wifi 进行连接控制，有效减少了实验有线的连接，使得实验操作更加便捷、简单，为物联实验增加了更多创意的可能。

2. 提供基本的数字和模拟端口的读写操作

基于无线连接，提供了 digitalRead/Write，analogRead/Write 等基本 I/O 操作的命令。在这个基础上可以完成很多基本的实验，比如：人体红外传感器、土壤湿度传感器、继电器的控制等。

3. 提供了常用的设备操控支持

目前已经支持的设备有：DHT11 温湿度传感器、舵机、LED2812、TCS230 颜色传感器，后续可以根据需求进行扩展。

4. 通过电脑命令进行无线控制

由于固件提供了基本 I/O 的读写操作，同时提供了常用模块的操作支持。只需要烧写一次固件，后续可以用电脑命令进行控制和数据的读取，避免了反复修改、烧录固件程序，降低了使用难度，使得能够专注于实验和创意本身。

平台命令说明

准备工作

使用附件中的”NetAssist”作为调试工具，协议类型选择“TCP Server”，端口设置为 1180，并点击打开。

固件对应的代码修改，包括 Wifi 的名称和密码，以及上面的主机地址。

```
const char* ssid     = "WifiName";          // Wifi 的名称
const char* password = "12345678";          // Wifi 的密码
const char* host = "192.168.31.248";
```

代码修改后编译更新模块，接下来给模块上电后就可以透过“NetAssist”给模块下命令。具体命令以及使用的方法可以参考后续内容。

1. 测试命令

命令：CMD:Beat;
返回值：
ACK:OK
功能：用于测试设备是否已经正常连接

2. 获取设备 MAC 地址

命令：CMD:Mac;
返回值：
ACK:MAC 地址
ACK:OK
例如：
ACK:60:01:94:51:E1:E5
ACK:OK
功能：获取设备 MAC 地址。

3. 端口模式设置命令

命令：CMD:pinMode,pin,input/output;
返回值：
ACK:OK
功能：用于设置设备第 pin 个 I/O 引脚为 input 或 output 模式。
例子：
CMD:pinMode,16,output;
设置第 16 脚为输出脚。
返回 ACK:OK 说明设置成功。

4. 数字口读命令

命令：CMD:digitalRead,pin;
返回值：
RET:0/1 // 返回当前的值 0 或 1
ACK:OK
功能：读取数字输入口的值
例子：
CMD:pinMode,0,input;
CMD:digitalRead,0;
返回：

RET:1
ACK:OK

5. 数字口写命令

命令：CMD:digitalWrite,pin,value;
返回值：ACK:OK
功能：向第 pin 个数字脚写入 value 值。
例子：
CMD:pinMode,2,output;
CMD:digitalWrite,2,1;
将第 2 个 IO 设置为输出，并设置为高电平输出。

6. 模拟口读命令

命令：CMD:analogRead,pin;
返回值：
RET:value
ACK:OK
功能：读取第 pin 个模拟口的数值。
例子：
CMD:analogRead,0;
读取模拟 0 口的数值，返回：
RET:1024
ACK:OK

7. 模拟口写命令

命令：CMD:analogWrite,pin,value;
返回值：ACK:OK
功能：向 pin 模拟口写入数值 value

8. LED2812 控制命令

命令：CMD:LED2812,pin,N,{i,r,g,b}.....;
返回值：ACK:OK
功能：设置第 pin 脚为 LED2812 的输出口，设定 N 个 LED 灯，后续通过{i,r,g,b}指定第 i 个灯的颜色为[r,g,b]。
例子：第 2 个 IO 为 LED2812 输出数字脚，有 8 个 LED 灯，每个灯有自己的颜色。
CMD:LED2812,2,8,{0,128,0,0},{1,255,201,14},{2,255,242,0},{3,0,128,0},{4,0,0,128},{5,181,230,29},{6,175,81,176},{7,255,174,201};
返回：ACK:OK

9. Server 控制命令

命令：CMD:Server,pin,degree;
返回值：ACK:OK
功能：通过 pin 脚控制舵机输出度数为 degree。
例子：通过第 12 脚控制舵机，控制角度为 45°
CMD:Server,12,45;
返回
ACK:OK

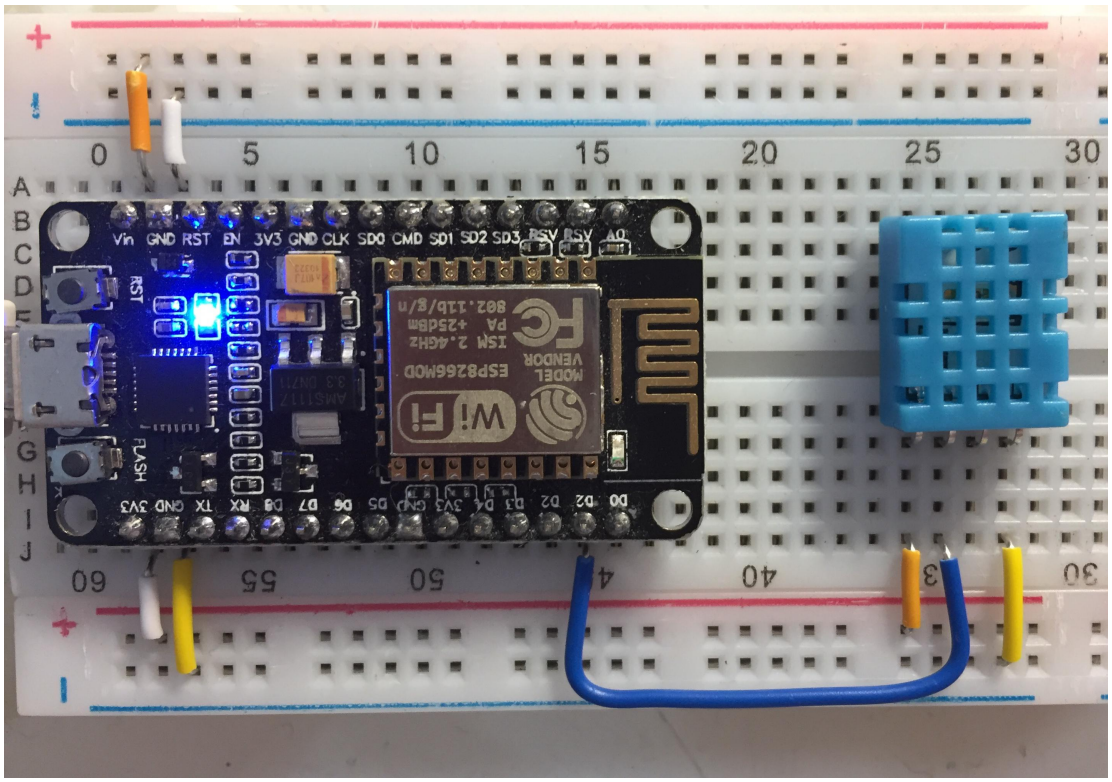
10. DHT11 控制命令

命令：CMD:DHT11,pin;
返回值：
RET:温度,湿度
ACK:OK
功能：通过 pin 脚与 DHT11 相连，并读取温度和湿度的数值。
例子：第 5 个 IO 与 DHT11 信号脚相连，读取温度和湿度数值：
CMD:DHT11,5;
返回
RET:29,68 // 温度 29°，湿度 68%
ACK:OK

平台实验示例

DHT11 温湿度传感器

正确连接 DHT11 的电源 VCC 与地 GND，将 ESP8266 模块的 GPIO5(D1)与 DHT11 信号线连接。参考如下图：



在 NetAssist 输入命令:

CMD:DHT11,5;

可以得到类似如下的返回值：

RET:28,50

ACK:OK


网络调试助手
网络设置
（1）协议类型
TCP Server
（2）本地主机地址
192.168.31.248
（3）本地主机端口
1180
关闭
接收区设置
接收转向文件...
自动换行显示
显示接收时间
十六进制显示
暂停接收显示
保存数据
清除接收
发送区设置
启用文件数据源...
自动发送附加位
自动发送应答包...
按十六进制发送
发送周期
5000
ms
文件载入
清除输入
网络数据接收
@野人 V4.2.5
【Receive from 192.168.31.8 :4097】:
RET:28,50
ACK:OK
客户端:
All Connections
断开
CMD:DHT11,5;
发送
接收区设置
TX:2916
RX:1933
复位计数

源代码

```
/*
 * 名称：多功能无线物联实验平台
 * 作者：周米笑
 * 邮件：mixiao.zhou@qq.com
 * Github: https://github.com/mixiao07/wirelessio
 * 版本：2018-05-01 1.0.0
 * 备注：
 *      材料：
 *      ESP8266
 *      LED2812
 *      继电器
 *      Server
 *      人体传感器
 *      其他：面包板、杜邦线、电源等
 *
 *      开发工具：
 *      Arduino 1.8.1
 */

#include <ESP8266WiFi.h>
#include <Servo.h>
#include <dht11.h>
#include <ColorRecognition.h>
#include <ColorRecognitionTCS230PI.h>
#include "Adafruit_NeoPixel.h"

const char* ssid          = "Xiaomi_1201";
const char* password = "19850322";
const char* host = "192.168.31.248";
const int hostPort = 1180;

void setup() {
   Serial.begin(115200);
   delay(10);
   pinMode(16, OUTPUT);    // LED pin

   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);

   WiFi.begin(ssid, password);
```

```
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

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

bool isConnected = false;
int    disCount = 0;
int    cycle = 20;
WiFiClient client;
bool cmd_process(const char *pcmd);

void loop() {
    if (!isConnected) {
        Serial.print("connecting to ");
        Serial.print(host);
        Serial.print(" port ");
        Serial.println(hostPort);
        if (!client.connect(host, hostPort)) {
            Serial.println("connection failed");
            delay(1000);
            return;
        }
        isConnected = true;
        disCount = 0;
        Serial.println("connect to host ok!");
    }

    if (!client.connected()) {
        disCount++;
        if (disCount > 100) {
            isConnected = false;
            Serial.println("not connected, re-connect");
        }
        delay(30);
        return;
    }
```

```
    while(client.available()){
        char *pcmd;
        String cmd = client.readStringUntil(';');
        pcmd = (char*)cmd.c_str();
        if(strncmp(pcmd, "CMD:", 4) != 0) {
            continue;
        }
        Serial.print("Get command from host: ");
        Serial.println(pcmd);
        if (cmd_process(pcmd+4)) {
            client.print("ACK:OK\r\n");
        }
        else {
            client.print("ACK:NG\r\n");
        }
    }

    delay(cycle);
}

bool isStrContainX(const char* px, const char *pstr)
{
    if (px == NULL || pstr == NULL)
        return false;
    if (strncmp(px, pstr, strlen(px)) == 0)
        return true;
    return false;
}

#define CMD_BEAT                        ("Beat")
#define CMD_MAC                          ("Mac")
#define CMD_PINMODE                      ("pinMode")
#define CMD_DIGITAL_READ                ("digitalRead")
#define CMD_DIGITAL_WRITE               ("digitalWrite")
#define CMD_ANALOG_READ                  ("analogRead")
#define CMD_ANALOG_WRITE                 ("analogWrite")

#define CMD_LED2812                      ("LED2812")
#define CMD_SERVER                       ("Server")
#define CMD_COLOR                        ("Color")
#define CMD_DHT11                        ("DHT11")


#define IS_CMD_X(x,cmd)                 (isStrContainX(x, cmd))
```

```
bool cmd_led2812_process(const char *pcmd);
bool cmd_server_process(const char *pcmd);
bool cmd_color_process(const char* pcmd);
bool cmd_dht11_process(const char *pcmd);

bool cmd_process(const char *pcmd)
{
   if (IS_CMD_X(CMD_BEAT, pcmd)
   {
      return true;
   }
   else if (IS_CMD_X(CMD_MAC, pcmd)) {
      String msg = "ACK:";
      msg += WiFi.macAddress();
      msg += "\r\n";
      client.print(msg);
      Serial.println(msg);
      return true;
   }
   else if (IS_CMD_X(CMD_PINMODE, pcmd)) {
      // CMD:pinMode,x,output/input
      pcmd+=strlen(CMD_PINMODE)+1;
      int pin=atoi(pcmd);
      if(pin>32)
      {
         Serial.println("Invalid pin > 9");
         return false;
      }

      pcmd=strpbrk(pcmd, ",");
      if(pcmd==NULL)
      {
            Serial.println("Invalid pcmd without mode");
            return false;
      }
      pcmd++;
      int mode;

      if(isStrContainX("input", pcmd))
      {
         mode=INPUT;
      }
      else if(isStrContainX("output", pcmd))
```

```
    {
      mode=OUTPUT;
    }
    else
    {
      Serial.println("Invalid mode");
      return false;
    }
    pinMode(pin,mode);

    String log="pinMode(";
    log += pin;
    log +=    ", ";
    log += mode;
    log += ")";
    Serial.println(log);
    return true;
  }
  else if (IS_CMD_X(CMD_DIGITAL_READ, pcmd)) {
    pcmd+=strlen(CMD_DIGITAL_READ)+1;
    int pin;
    pin=atoi(pcmd);
    if(pin>32)
    {
      Serial.println("Invalid digitalRead value");
      return false;
    }
    int value=digitalRead(pin);
    String log="digitalRead(";
    log+=pin;
    log+=") = ";
    log+=value;
    Serial.println(log);
    String msg = "RET:";
    msg += value;
    msg += "\r\n";
    client.print(msg);
    return true;
  }
  else if(IS_CMD_X(CMD_DIGITAL_WRITE,pcmd)) {
    // CMD:digitalWrite,pin,value (0:low, 1:high)
    pcmd+=strlen(CMD_DIGITAL_WRITE)+1;

    int pin;
```

```
    pin=atoi(pcmd);
    if(pin>32)
    {
       Serial.println("Invalid pin value");
       return false;
    }

    int value;
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL)
    {
       Serial.println("Invalid CMD_DIGITAL_WRITE value");
       return false;
    }
    pcmd++;
    value=atoi(pcmd);
    if(value==0)
    {
       value=LOW;
    }
    else if(value==1)
    {
       value=HIGH;
    }
    else
    {
       Serial.printf("Invalid value");
       return false;
    }
    digitalWrite(pin, value);
    String log="digitalWrite(";
    log += pin;
    log += ", ";
    log += value;
    log += ")";
    Serial.println(log);
    return true;
  }
  else if(IS_CMD_X(CMD_ANALOG_READ,pcmd)){
    // CMD:analogRead,x
    pcmd+=strlen(CMD_ANALOG_READ)+1;
    int pin;
    pin=atoi(pcmd);
    if(pin!=0)
```

```
    {
      Serial.println("Invalid analogRead value");
      return false;
    }
    pinMode(A0, INPUT);
    int value=analogRead(A0);
    String log="analogRead(";
    log += A0;
    log += ") = ";
    log += value;
    Serial.println(log);
    String msg = "RET:";
    msg += value;
    msg += "\r\n";
    client.print(msg);
    return true;
  }
  else if(IS_CMD_X(CMD_ANALOG_WRITE,pcmd)){
    // CMD:pin,value;
    int pin = atoi(pcmd);
    if (pin > 32)
    {
      return false;
    }
    pcmd = strpbrk(pcmd, ",");
    if (pcmd == NULL)
    {
      return false;
    }
    pcmd++;
    int value = atoi(pcmd);
    if (value > 1024)    {
      return false;
    }
    analogWrite(pin, value);
    String log = "analogWrite(";
    log += pin;
    log += ", ";
    log += value;
    log += ");";
    Serial.println(log);
    return true;
  }
  else if (IS_CMD_X(CMD_LED2812,pcmd)) {
```

```
        pcmd += strlen(CMD_LED2812) + 1;
        return cmd_led2812_process(pcmd);
    }
    else if (IS_CMD_X(CMD_SERVER,pcmd)) {
        pcmd += strlen(CMD_SERVER) + 1;
        return cmd_server_process(pcmd);
    }
    else if (IS_CMD_X(CMD_COLOR,pcmd)) {
        pcmd += strlen(CMD_COLOR) + 1;
        return cmd_color_process(pcmd);
    }
    else if (IS_CMD_X(CMD_DHT11,pcmd)) {
        pcmd += strlen(CMD_DHT11) + 1;
        return cmd_dht11_process(pcmd);
    }

    return false;
}


Adafruit_NeoPixel *pled2812=NULL;
bool cmd_led2812_process(const char *pcmd)
{
    // CMD:led2812,pin,N,{i,r,g,b}....;
    int pin=atoi(pcmd);
    if(pin>32)
    {
        Serial.println("Invalid pin value");
        return false;
    }
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL)
    {
        return false;
    }
    int n;
    pcmd++;
    n=atoi(pcmd);
    if(n<=0)
    {
        Serial.println("Invalid n value");
        return false;
    }
```

```
if (pled2812 == NULL) {
    pled2812 = new Adafruit_NeoPixel(n, pin, NEO_GRB + NEO_KHZ800);
    pled2812->begin();
}
else if ((pled2812->numPixels() != n) || (pled2812->getPin() != pin)) {
    delete pled2812;
    pled2812 = new Adafruit_NeoPixel(n, pin, NEO_GRB + NEO_KHZ800);
    pled2812->begin();
}

while(true) {
    pcmd = strpbrk(pcmd, "{");
    if(pcmd==NULL)
    {
        break;
    }
    int i;
    pcmd++;
    i=atoi(pcmd);
    if(i>=n || i<0)
    {
        continue;
    }
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL) continue;
    pcmd++;
    int r;
    r=atoi(pcmd);
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL) continue;
    pcmd++;
    int g;
    g=atoi(pcmd);
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL) continue;
    pcmd++;
    int b;
    b=atoi(pcmd);

    uint32_t color = pled2812->Color(r, g, b);
    pled2812->setPixelColor(i, color);
}
pled2812->show();
```

```
    Serial.println("led2812 show");
    return true;
}

Servo server_x;
bool cmd_server_process(const char *pcmd)
{
    // CMD:Server,pin,degree
    int pin=atoi(pcmd);
    if(pin>32)
    {
        Serial.println("Invalid pin value");
        return false;
    }
    int degree;
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL)
    {
        return false;
    }
    pcmd++;
    degree=atoi(pcmd);
    if(degree<0 || degree>180)
    {
        Serial.println("Invalid degree value");
        return false;
    }
    server_x.attach(pin);
    server_x.write(degree);
    delay(100);
    Serial.println("server run");
    return true;
}

ColorRecognitionTCS230PI *ptcs230 = NULL;
bool cmd_color_process(const char* pcmd)
{
    if (isStrContainX("pin", pcmd)) {
        // CMD:Color,pin,out,s2,s3
        pcmd += strlen("pin") + 1;

        int out;
        out=atoi(pcmd);
        if(out>32) {
```

```
        Serial.println("Invalitd out value");
        return false;
    }
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL)
    {
        return false;
    }
    pcmd++;
    int s2;
    s2=atoi(pcmd);
    if(s2>32)
    {
        Serial.println("Invalitd s2 value");
        return false;
    }
    pcmd=strpbrk(pcmd,",");
    if(pcmd==NULL)
    {
        return false;
    }
    pcmd++;
    int s3;
    s3=atoi(pcmd);
    if(s3>32)
    {
        Serial.println("Invalitd s3 value");
        return false;
    }

    if (ptcs230 != NULL) {
        delete ptcs230;
        ptcs230 = NULL;
    }
    ptcs230 = new ColorRecognitionTCS230PI(out, s2, s3);
    if (ptcs230 != NULL) {
        Serial.println("New Color Recognition OK");
    }
    else {
        Serial.println("New Color Recognition Fail");
        return false;
    }
}
else if (isStrContainX("whiteBalance", pcmd)) {
```

```
    //CMD:Color,whiteBalance;
    if (ptcs230 != NULL) {
        Serial.println("Start WhiteBalance");
        ptcs230->adjustWhiteBalance();
    }
    else {
        Serial.println("Invalid ptcs230");
        return false;
    }
}
else if (isStrContainX("balckBalance", pcmd)) {
    //CMD:Color,balckBalance;
    if (ptcs230 != NULL) {
        Serial.println("Start BlackBalance");
        ptcs230->adjustBlackBalance();
    }
    else {
        Serial.println("Invalid ptcs230");
        return false;
    }
}
else if (isStrContainX("colorRead", pcmd)) {
    //CMD:colorRead;
    if (ptcs230 == NULL) {
        Serial.println("Invalid ptcs230");
        return false;
    }

    int r, g, b;
    r = ptcs230->getRed();
    g = ptcs230->getGreen();
    b = ptcs230->getBlue();

    String msg = "RET:";
    msg += r;
    msg += ",";
    msg += g;
    msg += ",";
    msg += b;
    msg += ";\r\n";
    Serial.println(msg);
    client.print(msg);
}
else {
```

```
        return false;
    }

    return true;
}

bool cmd_dht11_process(const char *pcmd)
{
    // CMD:DHT11,pin;
    int pin = atoi(pcmd);
    if (pin > 32)
    {
        Serial.println("Invalid pin value");
        return false;
    }

    dht11 DHT11;
    int ret = DHT11.read(pin);
    if (DHTLIB_OK != ret)
    {
        Serial.print("DHT11.read fail: ");
        Serial.println(ret);
        return false;
    }
    String msg = "RET:";
    msg += DHT11.temperature;
    msg += ",";
    msg += DHT11.humidity;
    msg += "\r\n";
    client.print(msg);
    Serial.println(msg);

    return true;
}
```