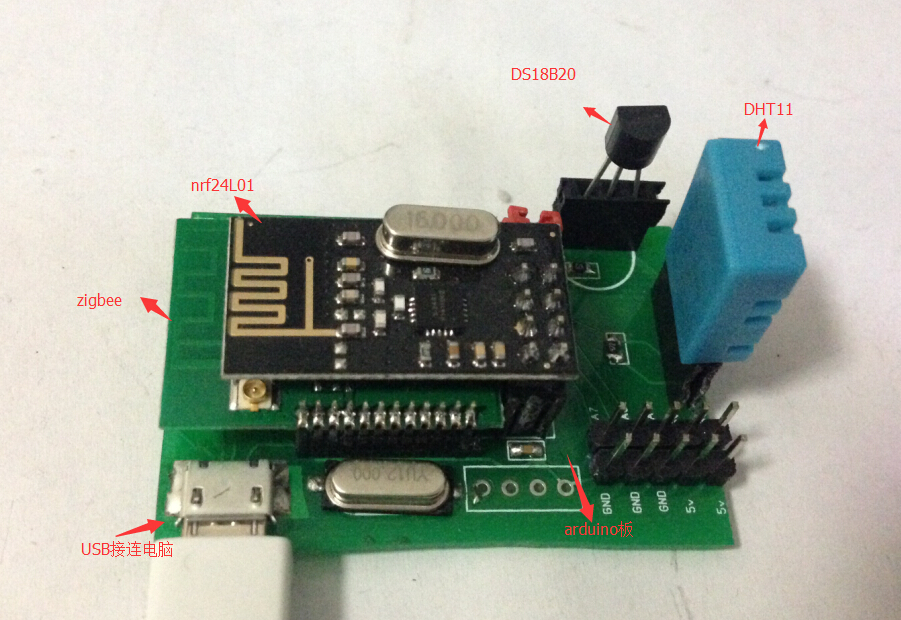
上篇[WIFI作品DIY教程07-《wifi家居网关》家居控制中心zigbee/nrf24l01等](http://www.znck007.com/forum.php?mod=viewthread&tid=24216)完成了家居网关。

温度湿度的场景用的很多，我们之前的教程也写过非常多的实例，

这次我们采用DS18B20获取温度，DHT11获取湿度，然后传给网关和云端保存数据，供我们观察分析等。

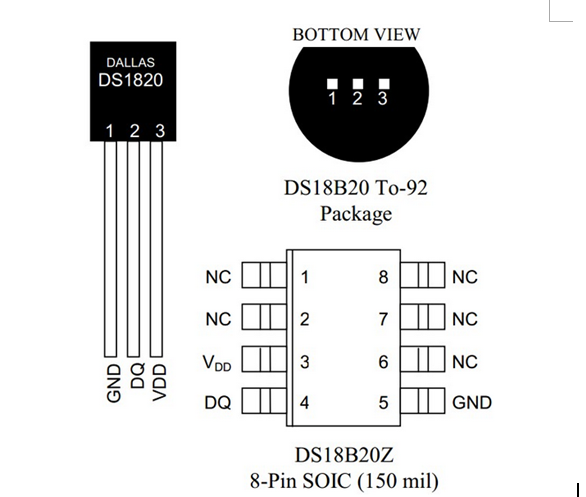


我们需要的配件：温湿传感器板、DS18B20、DHT11、nrf24l01（或zigbee等）

第一部分：DS18B20温度传感器介绍

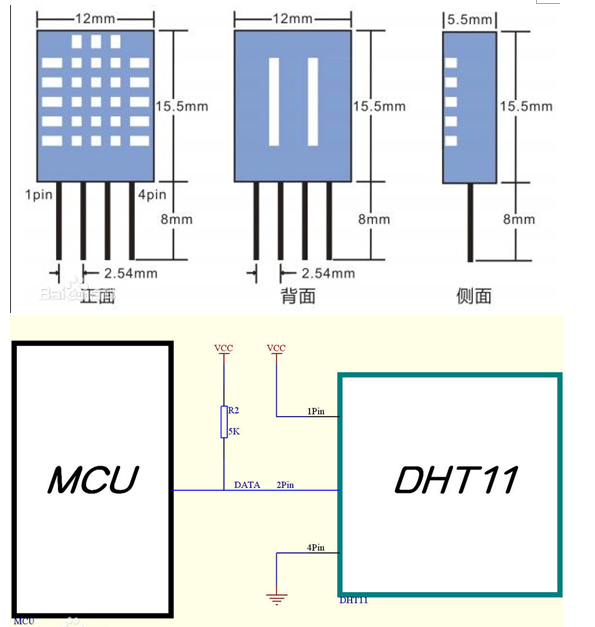
DS18B20是常用的温度传感器，具有体积小，硬件开销低，抗干扰能力强，精度高的特点。

DS18B20的数字温度计提供9至12位（可编程设备温度读数)。由于DS18B20是一条口线通信，所以中央微处理器与DS18B20只有一个一条口线连接。为读写以及温度转换可以从数据线本身获得能量，不需要外接电源。 因为每一个DS18B20的包含一个独特的序号，多个ds18b20s可以同时存在于一条总线。这使得温度传感器放置在许多不同的地方。它的用途很多，包括空调环境控制，感测建筑物内温设备或机器，并进行过程监测和控制。



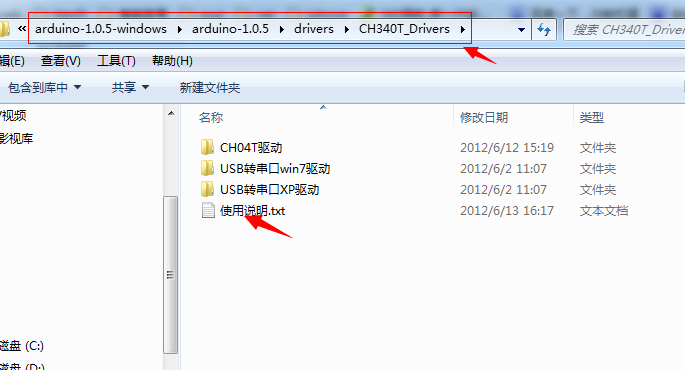
第二部分：DHT11湿度传感器介绍

DHT11数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器，它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性和卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个NTC测温元件，并与一个高性能8位单片机相连接。因此该产品具有品质卓越、超快响应、抗干扰能力强、性价比极高等优点。每个DHT11传感器都在极为精确的湿度校验室中进行校准。校准系数以程序的形式存在OTP内存中，传感器内部在检测型号的处理过程中要调用这些校准系数。单线制串行接口，使系统集成变得简易快捷。超小的体积、极低的功耗，使其成为该类应用中，在苛刻应用场合的最佳选择。



第三部分：下载编译源代码

1、安装CH340驱动



在drivers目录里找到CH340驱动，然后按说明安装（之前安装了就不用再安装了）。

2、源代码

/\*

\*NRF24l01针脚连接线

\* MISO -> 12

\* MOSI -> 11

\* SCK -> 13

\* Configurable:

\* CE -> 8

\* CSN -> 7

\*/

#include <SPI.h>

#include <Mirf.h>

#include <nRF24L01.h>

#include <MirfHardwareSpiDriver.h>

#include <OneWire.h>

int sid=1;//模块类型

int nid=2;//模块编号

//声名变量

const unsigned long interval = 60000;

unsigned long last\_sent;

OneWire ds(A5);

float celsius=0.0;

float fahrenheit=0.0;

int DHT11\_PIN=4;

int humdity=0;

int temperature=0;

//NRF24l01

unsigned long nrf24l01nowlast;

char nrf24l01buff[33]={0};

char nrf24l01Data;

int nrf24l01i=0;

void setup()

{

Serial.begin(115200);

char client[10]={0};//client

sprintf(client,"clie%d",sid);

//初始化Mirf，用于NRF24l01收发

Mirf\_Init(0,client,sid);

ini\_dht11();

Serial.println("zwifi\_wengshitu");

}

void loop()

{

unsigned long now = millis();

if ( now - last\_sent >= interval )

{

last\_sent = now;

celsius=0;

humdity=0;

run\_ds18();

if(celsius>0){

run\_dht11();

}

if(humdity>0){

char celstr[10]={0};

dtostrf(celsius,4,2,celstr);

char data[10]={0};

sprintf(data,"%d.%s",humdity,celstr);

send\_data(data);

}

}

}

//初始化Mirf 0初始化1为接收2为发送

void Mirf\_Init(int txrx,char \*server,int channel){

//初始化Mirf，用于NRF24l01收发

if(txrx==0) {

Mirf.spi = &MirfHardwareSpi;

Mirf.init();

Mirf.setRADDR((byte \*)server);//设置接收地址

}

if(txrx==1) {

Mirf.setRADDR((byte \*)server);//设置接收地址

}

if(txrx==2) {

Mirf.setTADDR((byte \*)server);//设置发送地址

}

Mirf.payload = sizeof(char);//收发字节

Mirf.channel = channel;

Mirf.config();

}

//NRF24l01发送函数

void Mirf\_Send(int channel,char \*server,char \*str){

Mirf\_Init(2,server,channel);

int bufi=0;

for(bufi=0;bufi<strlen(str);bufi++){//循环发送

char words=str[bufi];//发送的字符

Mirf.send((byte \*)&words);//发送命令

while(Mirf.isSending()){//等待发送完闭

}

delay(50);//延时，否则可能出现发送丢失现象

//Serial.print(words);

}

//Serial.println("");

}

void run\_ds18()

{

byte i;

byte present = 0;

byte type\_s;

byte data[12];

byte addr[8];

//float celsius, fahrenheit;

if ( !ds.search(addr)) {

//Serial.println("No more addresses.");

//Serial.println();

ds.reset\_search();

delay(250);

return;

}

//Serial.print("ROM =");

for( i = 0; i < 8; i++) {

//Serial.write(' ');

//Serial.print(addr[i], HEX);

}

if (OneWire::crc8(addr, 7) != addr[7]) {

Serial.println("CRC is not valid!");

return;

}

//Serial.println();

// the first ROM byte indicates which chip

switch (addr[0]) {

case 0x10:

Serial.print("Chip = DS18S20"); // or old DS1820

type\_s = 1;

break;

case 0x28:

Serial.print("Chip = DS18B20");

type\_s = 0;

break;

case 0x22:

Serial.print("Chip = DS1822");

type\_s = 0;

break;

default:

Serial.print("Device is not a DS18x20 family device.");

return;

}

ds.reset();

ds.select(addr);

ds.write(0x44,1); // start conversion, with parasite power on at the end

delay(1000); // maybe 750ms is enough, maybe not

// we might do a ds.depower() here, but the reset will take care of it.

present = ds.reset();

ds.select(addr);

ds.write(0xBE); // Read Scratchpad

//Serial.print(" Data = ");

//Serial.print(present,HEX);

//Serial.print(" ");

for ( i = 0; i < 9; i++) { // we need 9 bytes

data[i] = ds.read();

//Serial.print(data[i], HEX);

//Serial.print(" ");

}

//Serial.print(" CRC=");

//Serial.print(OneWire::crc8(data, 8), HEX);

//Serial.println();

// convert the data to actual temperature

unsigned int raw = (data[1] << 8) | data[0];

if (type\_s) {

raw = raw << 3; // 9 bit resolution default

if (data[7] == 0x10) {

// count remain gives full 12 bit resolution

raw = (raw & 0xFFF0) + 12 - data[6];

}

} else {

byte cfg = (data[4] & 0x60);

if (cfg == 0x00) raw = raw << 3; // 9 bit resolution, 93.75 ms

else if (cfg == 0x20) raw = raw << 2; // 10 bit res, 187.5 ms

else if (cfg == 0x40) raw = raw << 1; // 11 bit res, 375 ms

// default is 12 bit resolution, 750 ms conversion time

}

celsius = (float)raw / 16.0;

fahrenheit = celsius \* 1.8 + 32.0;

Serial.print(" Temperature = ");

Serial.print(celsius);

Serial.print(" Celsius, ");

Serial.print(fahrenheit);

Serial.println(" Fahrenheit");

}

void ini\_dht11()

{

DDRC|=\_BV(DHT11\_PIN);

PORTC|=\_BV(DHT11\_PIN);

}

byte read\_dht11\_dat()

{

byte i = 0;

byte result = 0;

for(i=0;i<8;i++)

{

while(!(PINC&\_BV(DHT11\_PIN)));

delayMicroseconds(30);

if(PINC&\_BV(DHT11\_PIN))

result|=(1<<(7-i));

while((PINC&\_BV(DHT11\_PIN)));

}

return result;

}

void run\_dht11()

{

byte dht11\_dat[5];

byte dht11\_in;

byte i;

PORTC &= ~\_BV(DHT11\_PIN);

delay(18);

PORTC|=\_BV(DHT11\_PIN);

delayMicroseconds(40);

DDRC &= ~\_BV(DHT11\_PIN);

delayMicroseconds(40);

dht11\_in = PINC & \_BV(DHT11\_PIN);

if(dht11\_in)

{

Serial.println("dht11 start condition 1 not met");

return;

}

delayMicroseconds(80);

dht11\_in=PINC & \_BV(DHT11\_PIN);

if(!dht11\_in)

{

Serial.println("dht11 start condition 2 not met");

return;

}

delayMicroseconds(80);

for(i=0;i<5;i++)

dht11\_dat[i]=read\_dht11\_dat();

DDRC|=\_BV(DHT11\_PIN);

PORTC|=\_BV(DHT11\_PIN);

byte dht11\_check\_sum = dht11\_dat[0]+dht11\_dat[1]+dht11\_dat[2]+dht11\_dat[3];

if(dht11\_dat[4]!=dht11\_check\_sum)

{

Serial.println("DHT11 checksum error");

}

Serial.print("Chip = DHT11 Current humdity= ");

Serial.print(dht11\_dat[0],DEC);

Serial.print(".");

Serial.print(dht11\_dat[1],DEC);

Serial.print("%");

Serial.print("temperature = ");

Serial.print(dht11\_dat[2],DEC);

Serial.print(".");

Serial.print(dht11\_dat[3],DEC);

Serial.println("C");

humdity=(int)dht11\_dat[0];

temperature=(int)dht11\_dat[2];

}

//

void send\_data(char \*data){

char server[10]={0};//server

sprintf(server,"serv%d",1);

//Serial.println(server);

char updateData[33]={0};

char front[10]={0};

//memcpy(front,body,9);

sprintf(front," {ck%03d%03d",sid,nid);

sprintf(updateData,"%s%s}",front,data);

Serial.println(updateData);

Serial.println();

Mirf\_Send(1,server,updateData);

char client[10]={0};//client

sprintf(client,"clie%d",sid);

Mirf\_Init(1,client,sid);

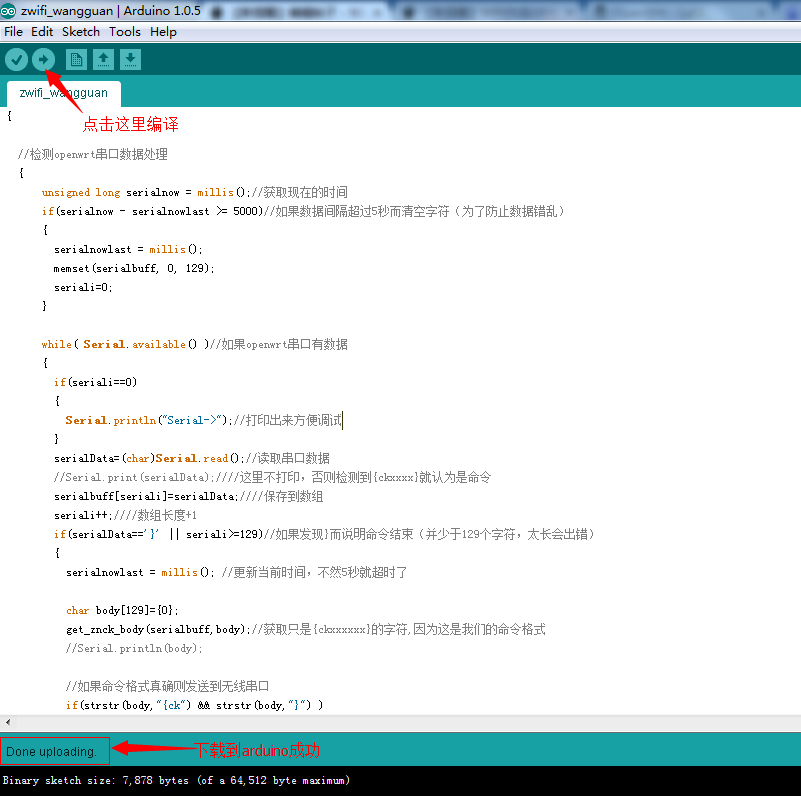
}

刷写arduino程序选哪个板？

请选择板是Arduino Uno，并记得选对对应的串口。

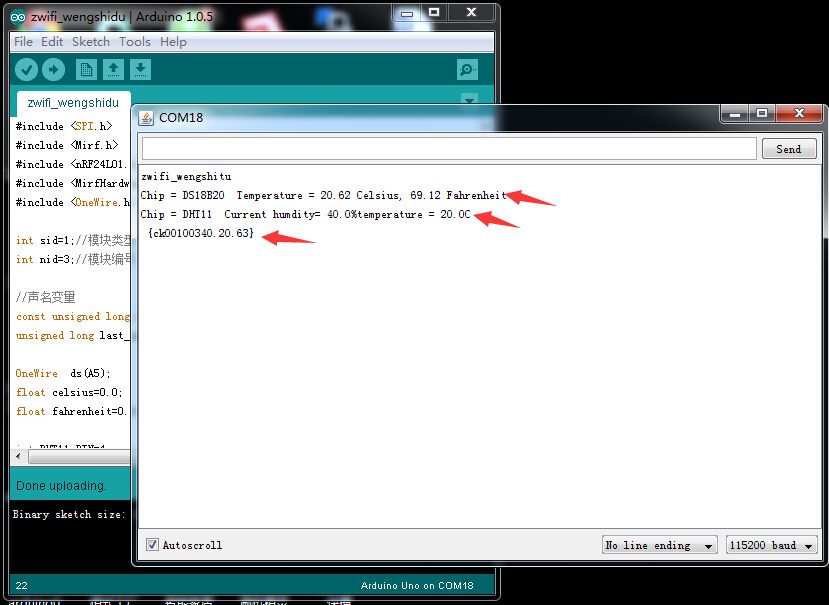
刷写ardruino源代码时，提示库不存在？

请联系qq610854837下载zwifi资料，软件下载目录里的arduino-1.0.5，里面包含了教程使用的所有库。



将代码进行编译下载到arduino板上。

3、串口运行查看数据是否正常



在arduino软件里，查看串口数据，记得是115200哦，等1分钟，有如上图，说明下载运行成功了！

程序的原理就是获取到二个传感器芯片的温温度，然后组合成字符{ck00x00xnnnn}发送到网关保存数据，供我们以后分析。

第四部分：检测通信和数据库记录

打开：<http://192.168.1.1/znck007/mysqltest.php?mode=select>



点击查看数据库记录，001，002的记录，说明网关成功保存了记录。30.20.50代表，湿度30%，温度是20.50度。

同时也可以用数据库软件Navicat for MySQL查看数据记录！

