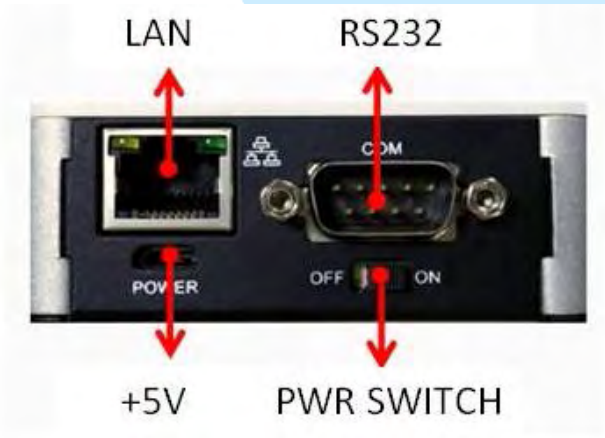
I.O.T

1. I.O.T Application

86Duino EduCake (EduCake) is an open-source microcomputer learning platform built on Vortex86EX, a System-On-Chip (SoC) built with a 32-bit x86 processor. The EduCake is designed to be electronically compatible with Arduino pin layouts by using an Arduino emulation software which enables application codes written for Arduino (sketch) to be compiled and run on the EduCake without modification. In a nutshell, the EduCake is hardware and software compatible to the Arduino platform. The EduCake is packaged in a compact, yet functional metallic enclosure with an integrated solderless breadboard that exposes the I/O, designed to help hobbyists and students experiment with different electronic circuitries, to learn microcomputer and embedded system, without the need to solder.

Unlike other Arduino learning kits, the EduCake does not need an additional W5100 LAN expansion board to have the LAN access. The EduCake has a build-in LAN module able to provide TCP/IP protocols (TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE and Ethernet). Also, the EduCake will only need to use Ethernet Library[1] to connect to internet.

(1) 86Duino EduCake



[1] Please refer Arduino.cc (http://www.arduino.cc/en/reference/ethernet) for more information.

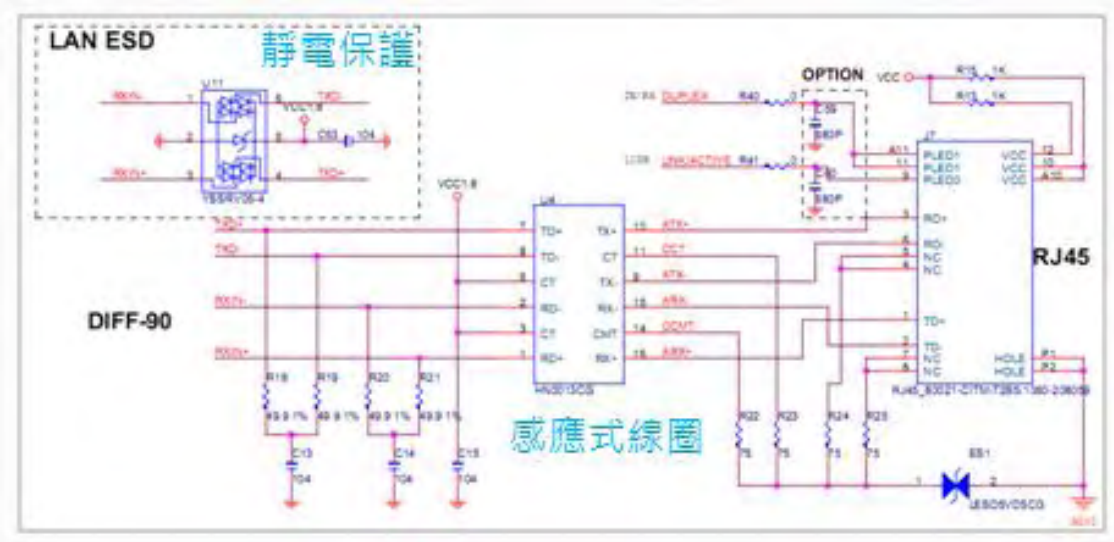
As the picture (154) shown, the RJ45 connector provides LAN accessibility.

As the pictures (a) (b) show, we can use the ordinary Cat 5 cable to connect to the EduCake and network switch.

(a). EduCake

(b). Network Switch

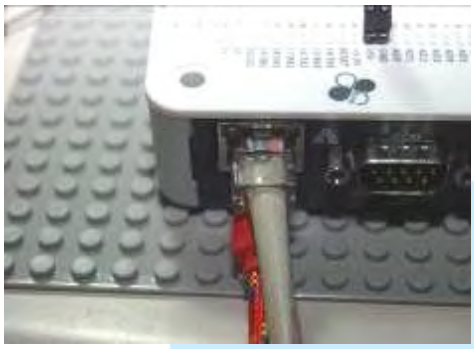
The EduCake provides LAN that supports up to 10/100Mbps. With ESD and induction coil designs, the Educake provides the proper ESD protection as well.



As the shown in the picture below, after booting up the EduCake, the green light to the right will stay on and the orange light to the left will blink.

2. The Simple Web Server Design

As seen in the pictures (a) (b) shown below, connect one end of Cat 5 cable to the Educake and the other end to the network switch.



(a). EduCake

(b). Network Switch

As with the previous chapters, after installing the 86Duino EduCake development software and opening the IDE, we can write a program (shown below) to display the value of AnalogPort0~Port5 from the Educake to the simple webserver we have designed.

WebServer Test Program

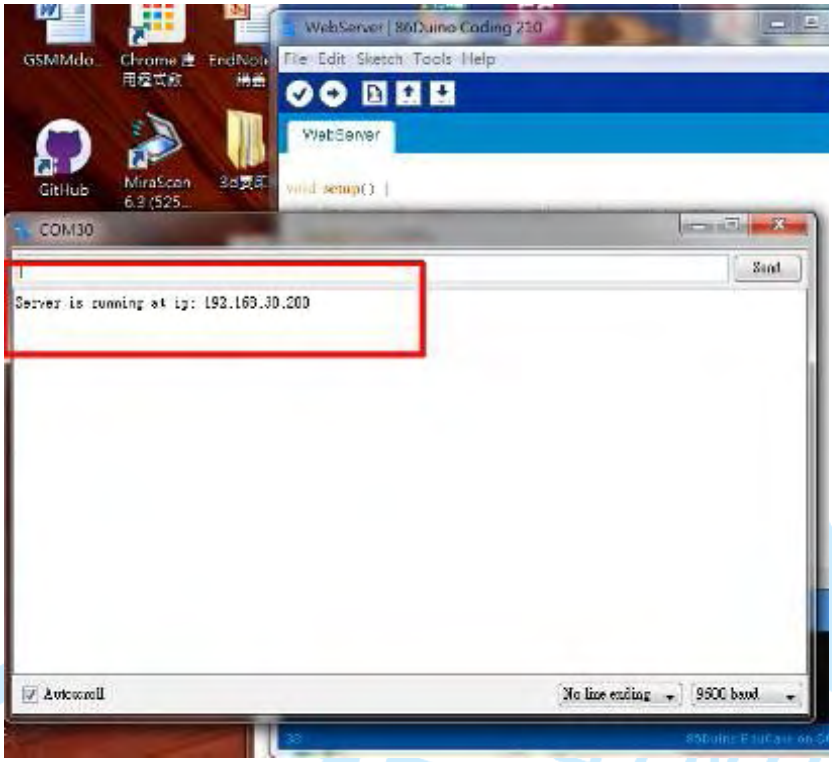
```
#include <SPI.h>
#include <Ethernet.h>
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
   0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);
// the router's gateway address: IPAddress gateway(192, 168, 30, 254);
// the subnet: IPAddress subnet(255, 255, 255, 0);


// Initialize the Ethernet server library
```

```
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("Server is running at ip: ");
  Serial.println(Ethernet.localIP());
}


void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("New Client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
```

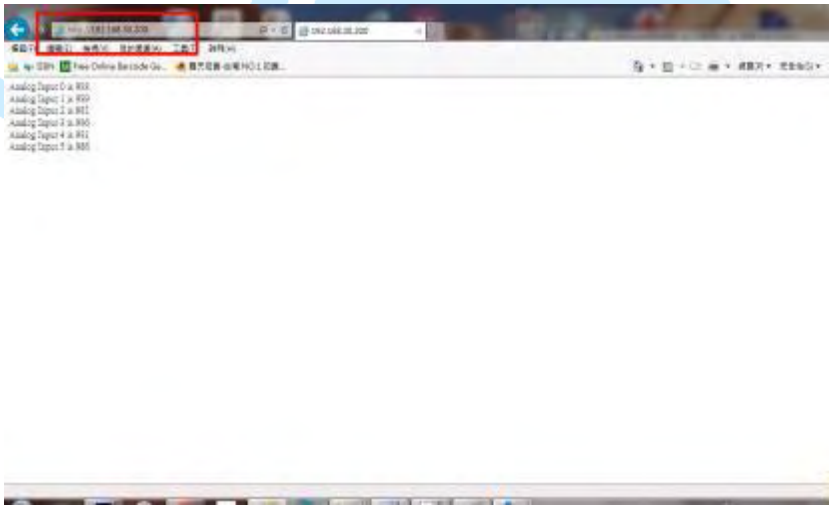
```
                // if you've gotten to the end of the line (received a newline
                // character) and the line is blank, the http request has ended,
                // so you can send a reply
                if (c == '\n' && currentLineIsBlank) {
                  // send a standard http response header
                  client.println("HTTP/1.1 200 OK");
                  client.println("Content-Type: text/html");
                  client.println("Connection: close");
                  client.println("Refresh: 5");
                  client.println();
                  client.println("<!DOCTYPE HTML>");
                  client.println("<html>");
                  // output the value of each analog input pin
                  for (int analogChannel = 0; analogChannel < 6; analog-
Channel++) {                    int sensorReading =
analogRead(analogChannel);
                    client.print("Analog Input ");
                    client.print(analogChannel);
                client.print(" is ");
                    client.print(sensorReading);
                    client.println("<br />");
                  }
                  client.println("</html>");
                  break;
                }
                if (c == '\n') {
                  // you're starting a new line
                  currentLineIsBlank = true;
                }
                else if (c != '\r') {
                  // you've gotten a character on the current line
```

```
                currentLineIsBlank = false;
            }
          }
        }
        // give the web browser time to receive the data
        delay(1);
        // close the connection:
        client.stop();
        Serial.println("Client Disconnected");
      }
    }
```

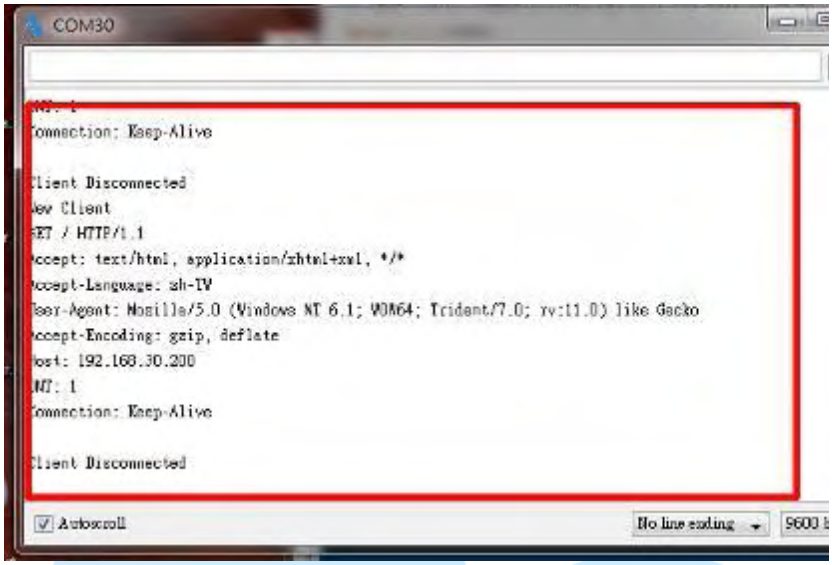
As shown in the pictures below, we can then see the test results from the Webserver testing program.



(a). WebServer Testing Program Start



(b). Connet the Webserver



(c). WebServer Testing Result

3.Use DHCP to Set up the Web Server

Using the previous setup with a Cat 5 Cable, the EduCake, and the Network Switch, we will use DHCP to set up the web server.

(a). EduCake

(b). Network Switch

As with the previous chapters, after installing the 86Duino EduCake development software and opening the IDE, we can write a program shown below to display the value of AnalogPort0~Port5 from the Educake to the simple webserver we have just designed. Unlike the previous section, however, the IP address is acquired through DHCP server.

WebServer →(WebServer_Dhcp)

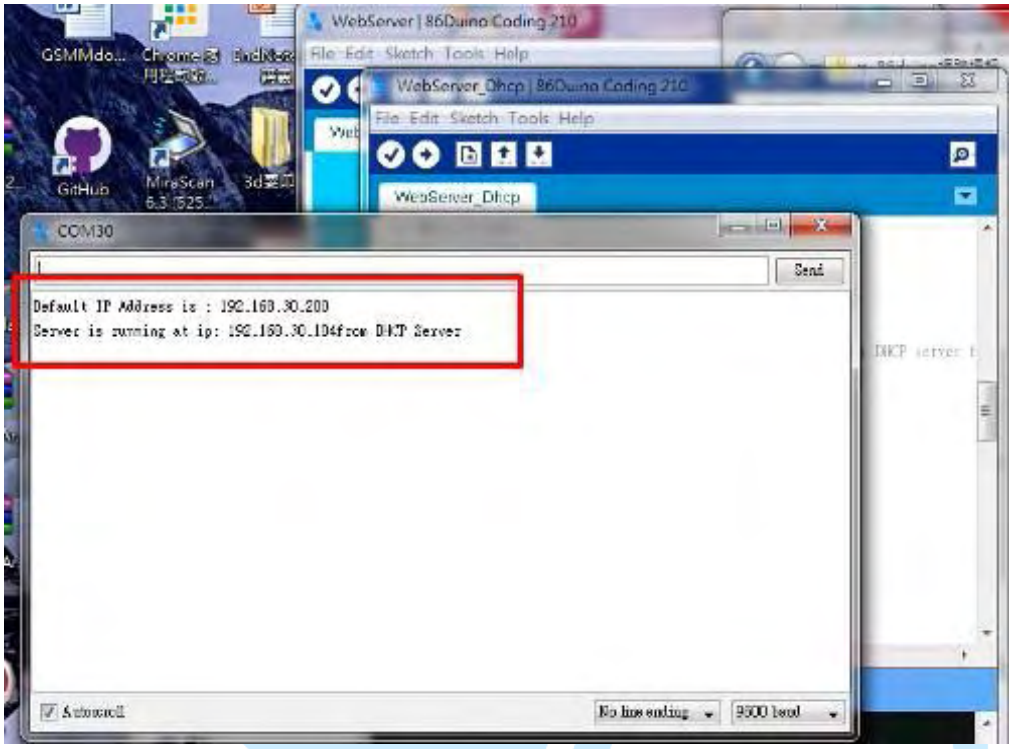
```
#include <SPI.h>
#include <Ethernet.h>
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);
// the subnet:
```

```
IPAddress subnet(255, 255, 255, 0);
// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
    Serial.println("Now Program Start") ;

  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

    if (Ethernet.begin(mac) == 0) {
    Serial.println("I can't get any    IP address from DHCP Server");
    for(;;)
      ;
  }

  Serial.print("Default IP Address is : ");
  Serial.println(ip);

Ethernet.begin(mac, ip, dnServer, gateway, subnet);
    Ethernet.begin(mac);
  server.begin();
  Serial.print("Server is running at ip: ");
  Serial.print(Ethernet.localIP());
  Serial.print("from DHCP Server \n");}
```

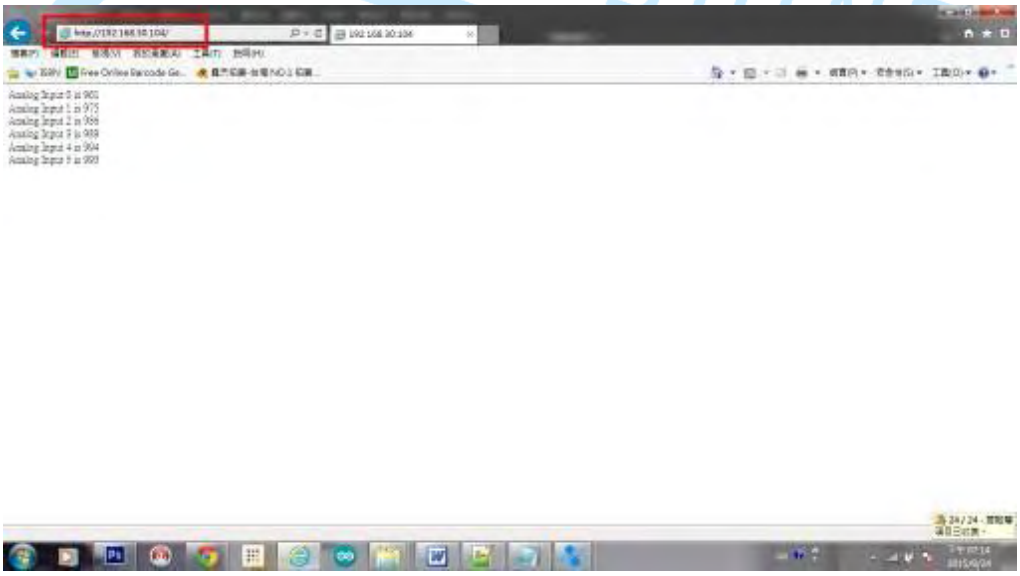
```
        void loop() {
          // listen for incoming clients
          EthernetClient client = server.available();
          if (client) {
            Serial.println("New Client");
            // an http request ends with a blank line
            boolean currentLineIsBlank = true;
            while (client.connected()) {
              if (client.available()) {
                char c = client.read();
                Serial.write(c);
                // if you've gotten to the end of the line (received a newline
                // character) and the line is blank, the http request has ended,
                // so you can send a reply
                if (c == '\n' && currentLineIsBlank) {
                  // send a standard http response header
                  client.println("HTTP/1.1 200 OK");
                  client.println("Content-Type: text/html");
        client.println("Connection: close");    // the connection will be closed after
completion of the response
            client.println("Refresh: 5");    // refresh the page automatically every
5 sec
                  client.println();
                  client.println("<!DOCTYPE HTML>");
                  client.println("<html>");
                  // output the value of each analog input pin
                  for (int analogChannel = 0; analogChannel < 6; analog-
Channel++) {
                    int sensorReading = analogRead(analogChannel);
                    client.print("Analog Input ");
                    client.print(analogChannel);
```

```
                client.print(" is ");
                client.print(sensorReading);
                client.println("<br />");
            }
            client.println("</html>");
            break;
          }
          if (c == '\n') {
            // you're starting a new line
            currentLineIsBlank = true;
          }
          else if (c != '\r') {
            // you've gotten a character on the current line
            currentLineIsBlank = false;
          }
        }
      }
      // give the web browser time to receive the data
      delay(1);
      // close the connection:
      client.stop();
      Serial.println("Client Disconnected");
    }
}
```

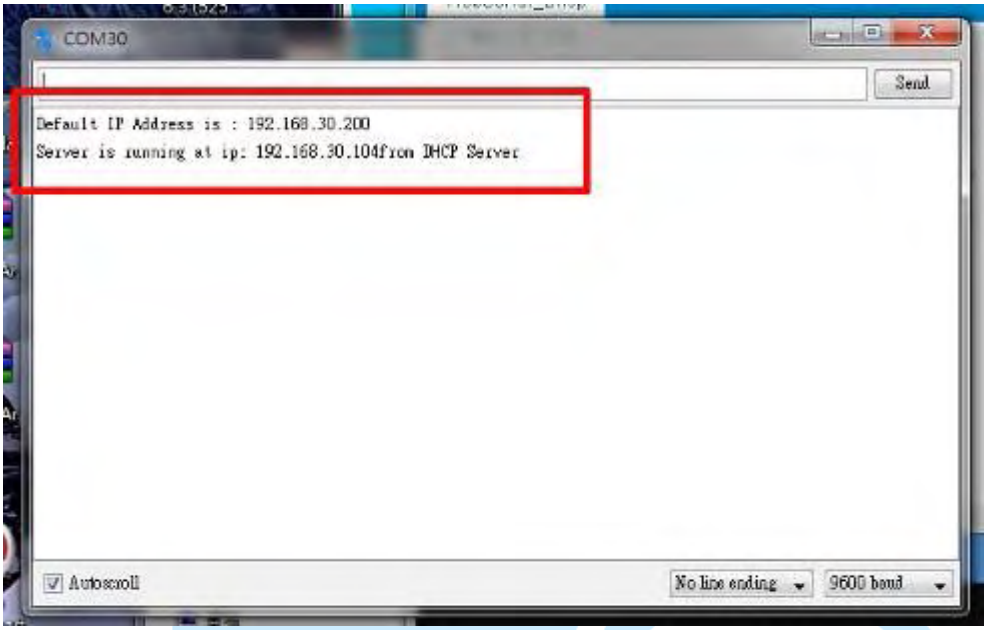
Below are the testing results.



(a). WebServer Testing Program Start



(b). Connet the Web Server



(c). WebServer Testing Result

4.Telnet Testing Program

We will continue using the same setup to test Telnet functionality.

(a). EduCake

(b). Network Switch

Once connected, the program below can be used to turn the Educake into a Telnet client platform.

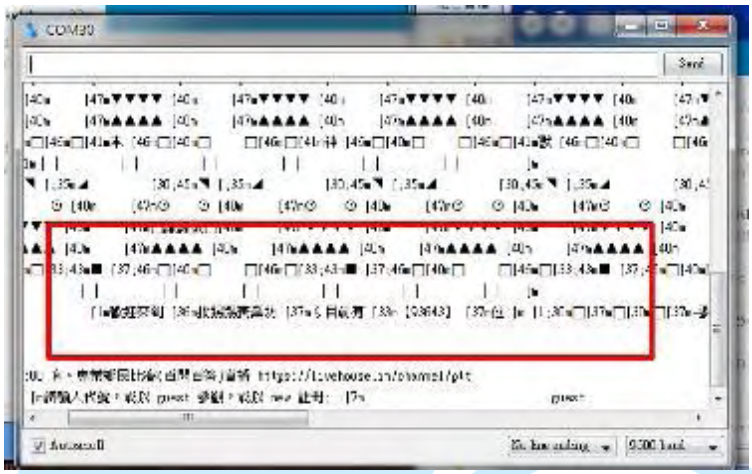
Telnet Client Testing Program

```
#include <SPI.h>
#include <Ethernet.h>
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
   0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
// MAC ADDRESS
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);
// the subnet:
IPAddress subnet(255, 255, 255, 0);
```

```
// Enter the IP address of the server you're connecting to:
IPAddress server(140, 112, 172, 11);

// Initialize the Ethernet client library
// with the IP address and port of the server
// that you want to connect to (port 23 is default for telnet;
// if you're using Processing's ChatServer, use  port 10002):
EthernetClient client;

void setup() {
  // start the Ethernet connection:
  Ethernet.begin(mac, ip, dnServer, gateway, subnet);
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  // give the Ethernet shield a second to initialize:
  delay(1000);
  Serial.println("connecting...");

  // if you get a connection, report back via serial:
  if (client.connect(server, 23)) {
    Serial.println("connected");
  }
  else {
    // if you didn't get a connection to the server:
    Serial.println("connection failed");
```

```
    }
}

void loop()
{
    // if there are incoming bytes available
    // from the server, read them and print them:
    if (client.available()) {
        char c = client.read();
        Serial.print(c);
    }

    // as long as there are bytes in the serial queue,
    // read them and send them out the socket if it's open:
    while (Serial.available() > 0) {
        char inChar = Serial.read();
        if (client.connected()) {
            client.print(inChar);
        }
    }

    // if the server's disconnected, stop the client:
    if (!client.connected()) {
        Serial.println();
        Serial.println("disconnecting.");
        client.stop();
        // do nothing:
        while (true);
    }
}
```

Below are the results of testing.



5.Text Web Client Browser Testing Program

(a). EduCake

(b). Network Switch

We will now use the same setup to turn the Educake into a text based web browser client to search for information on the internet.

Text Web Client Browser (WebClient)

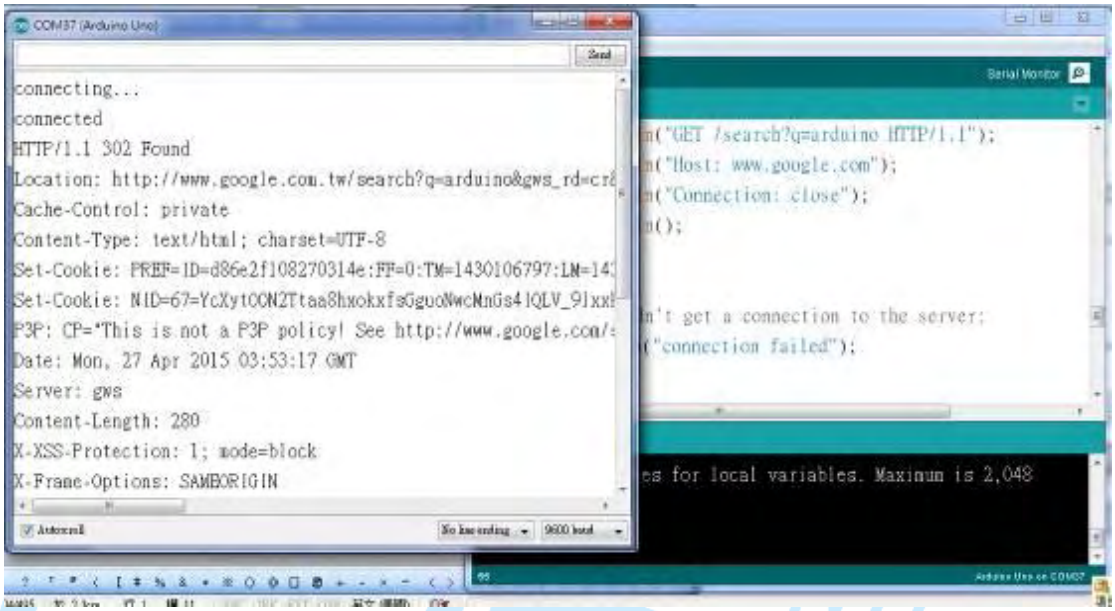
```
#include <SPI.h>
#include <Ethernet.h>



// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
byte mac[] = {
  0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);
// the subnet:
IPAddress subnet(255, 255, 255, 0);
```

```
        // if you don't want to use DNS (and reduce your sketch size)
        // use the numeric IP instead of the name for the server:
        //IPAddress server(74,125,232,128);    // numeric IP for Google (no
DNS)          //www.google.com
        char server[] = "www.google.com";       // name address for Google
(using DNS)


        // Initialize the Ethernet client library
        // with the IP address and port of the server
        // that you want to connect to (port 80 is default for HTTP):
        EthernetClient client;


        void setup() {
          // Open serial communications and wait for port to open:
          Serial.begin(9600);
          while (!Serial) {
            ; // wait for serial port to connect. Needed for Leonardo only
          }


          // start the Ethernet connection:
          if (Ethernet.begin(mac) == 0) {
            Serial.println("Failed to configure Ethernet using DHCP");

            // no point in carrying on, so do nothing forevermore:
            // try to congifure using IP address instead of DHCP:
              Ethernet.begin(mac, ip, dnServer, gateway, subnet);
          }
          // give the Ethernet shield a second to initialize:
          delay(1000);
          Serial.println("connecting...");
```

```
  // if you get a connection, report back via serial:
  if (client.connect(server, 80)) {
    Serial.println("connected");
    // Make a HTTP request:
   client.println("GET /search?q=arduino HTTP/1.1");
    client.println("Host: www.google.com");
    client.println("Connection: close");
    client.println();
  }
  else {
    // if you didn't get a connection to the server:
    Serial.println("connection failed");
  }
}

void loop()
{
  // if there are incoming bytes available
  // from the server, read them and print them:
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  // if the server's disconnected, stop the client:
  if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
```

```
        // do nothing forevermore:
        while (true);
    }
}
```

As the picture shows below, we can see the testing result.



6.Acquiring the Netwrok Protocol Time

Continuing as before with the pictures (a) (b) shown below, connect one end of Cat 5 cable to the Educake and the other end to the network switch.

(a). EduCake

(b). Network Switch

The following program will enable the Educake to a become web client able to acquire network protocol time.

UdpNtpClient

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
byte mac[] = {
  0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);      // the router's gateway address:
IPAddress gateway(192, 168, 30, 254);// the subnet:
IPAddress subnet(255, 255, 255, 0);
unsigned int localPort = 8888; // local port to listen for UDP packets
```

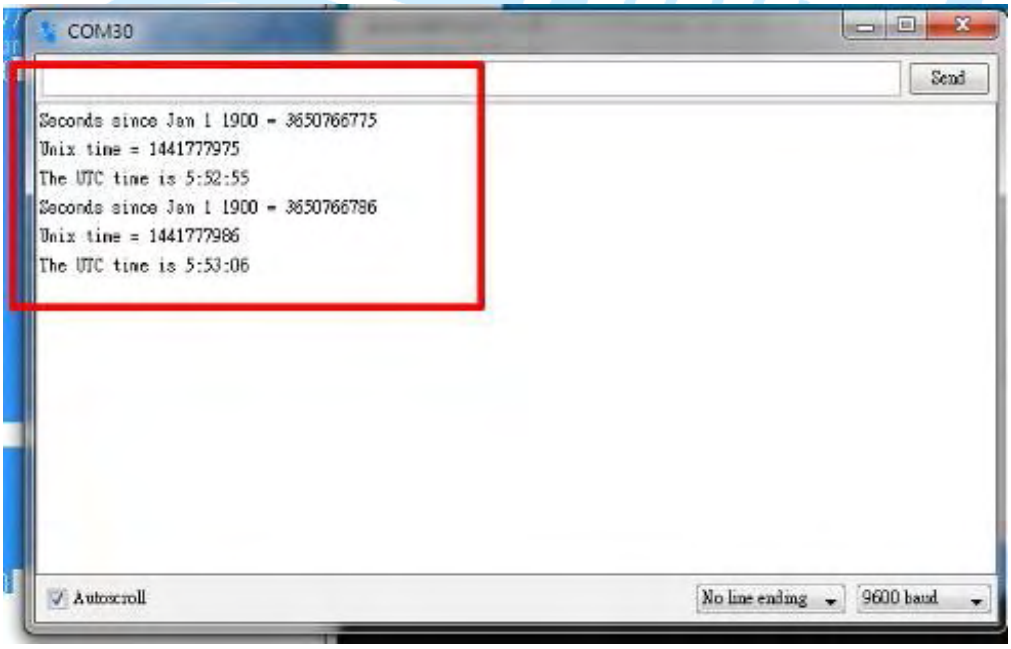
```
char timeServer[] = "time.nist.gov"; // time.nist.gov NTP server


const int NTP_PACKET_SIZE = 48;
byte packetBuffer[ NTP_PACKET_SIZE]; // A UDP instance to let us send and receive packets over UDP
EthernetUDP Udp;


void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }  // start Ethernet and UDP
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // no point in carrying on, so do nothing forevermore:
  Ethernet.begin(mac, ip, dnServer, gateway, subnet);
  }
  Udp.begin(localPort);
}


void loop()
{
  sendNTPpacket(timeServer); // send an NTP packet to a time server


  // wait to see if a reply is available
  delay(1000);
  if ( Udp.parsePacket() ) {
    // We've received a packet, read the data from it
```

```
        Udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the buffer
        //the timestamp starts at byte 40 of the received packet and is four bytes,
        // or two words, long. First, esxtract the two words:


        unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);

        unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);

        // combine the four bytes (two words) into a long integer
        // this is NTP time (seconds since Jan 1 1900):
        unsigned long secsSince1900 = highWord << 16 | lowWord;

        Serial.print("Seconds since Jan 1 1900 = " );
        Serial.println(secsSince1900);

        // now convert NTP time into everyday time:
        Serial.print("Unix time = ");
        // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
        const unsigned long seventyYears = 2208988800UL;
        // subtract seventy years:
        unsigned long epoch = secsSince1900 - seventyYears;
        Serial.println(epoch);



        // print the hour, minute and second:
        Serial.print("The UTC time is ");       // UTC is the time at Greenwich Meridian (GMT)
        Serial.print((epoch  % 86400L) / 3600); // print the hour (86400 equals secs per day)
        Serial.print(':');
```

```
            if ( ((epoch % 3600) / 60) < 10 ) {
              // In the first 10 minutes of each hour, we'll want a leading '0'
              Serial.print('0');
            }
            Serial.print((epoch  % 3600) / 60); // print the minute (3600 equals secs per minute)
            Serial.print(':');
            if ( (epoch % 60) < 10 ) {
              // In the first 10 seconds of each minute, we'll want a leading '0'
              Serial.print('0');
            }
            Serial.println(epoch % 60); // print the second
        }
        // wait ten seconds before asking for the time again
        delay(10000);
    }

    // send an NTP request to the time server at the given address
    unsigned long sendNTPpacket(char* address)
    {

        // set all bytes in the buffer to 0
        memset(packetBuffer, 0, NTP_PACKET_SIZE);
        // Initialize values needed to form NTP request
        // (see URL above for details on the packets)
        packetBuffer[0] = 0b11100011;   // LI, Version, Mode
        packetBuffer[1] = 0;        // Stratum, or type of clock
        packetBuffer[2] = 6;        // Polling Interval
        packetBuffer[3] = 0xEC;   // Peer Clock Precision
        // 8 bytes of zero for Root Delay & Root Dispersion
```

```
    packetBuffer[12]  = 49;
    packetBuffer[13]  = 0x4E;
    packetBuffer[14]  = 49;
    packetBuffer[15]  = 52;

    // all NTP fields have been given values, now
    // you can send a packet requesting a timestamp:
    Udp.beginPacket(address, 123); //NTP requests are to port 123
    Udp.write(packetBuffer, NTP_PACKET_SIZE);
    Udp.endPacket();
}
```

Below is the results of the network protocol time result.



7.Telnet Chat Room

(a). EduCake

(b). Network Switch

Using the same setup as all the other exercises with the Cat 5 cable, EduCake, and Network switch we'll be able to create a Telnet chat room server.

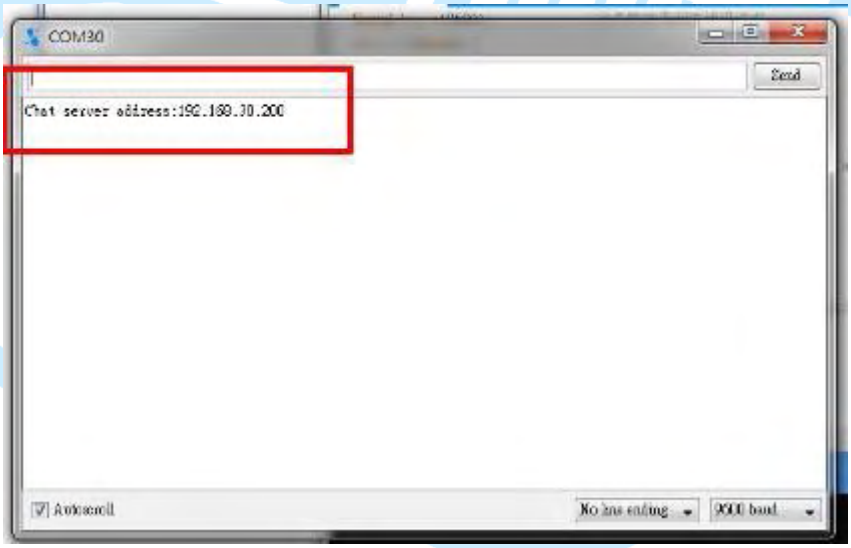
Telnet Chat Server

```
#include <SPI.h>
#include <Ethernet.h>


// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
byte mac[] = {
  0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);      // the router's gateway address:
IPAddress gateway(192, 168, 30, 254);// the subnet:
IPAddress subnet(255, 255, 255, 0);
// telnet defaults to port 23
EthernetServer server(23);
boolean alreadyConnected = false; // whether or not the client was connected previously
```

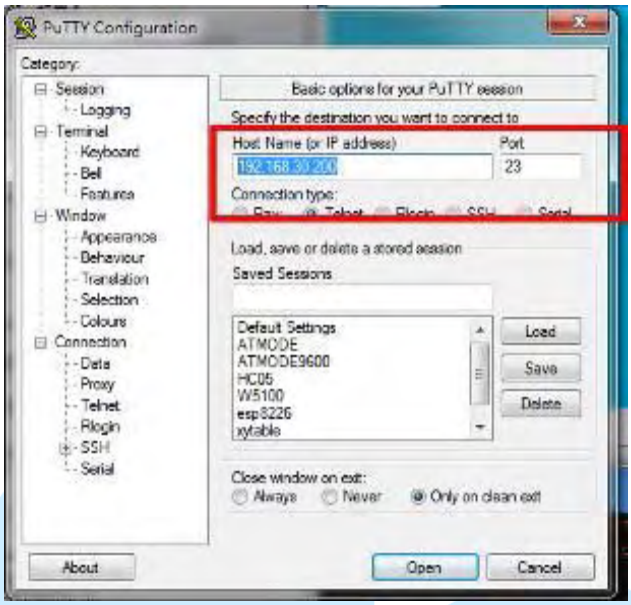
```
void setup() {
  // initialize the ethernet device
  Ethernet.begin(mac, ip, gateway, subnet);
  // start listening for clients
  server.begin();
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  Serial.print("Chat server address:");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // wait for a new client:
  EthernetClient client = server.available();


  // when the client sends the first byte, say hello:
  if (client) {
    if (!alreadyConnected) {
      // clead out the input buffer:
      client.flush();
      Serial.println("We have a new client");
      client.println("Hello, client!");
      alreadyConnected = true;

    }
```

```
        if (client.available() > 0) {
          // read the bytes incoming from the client:
          char thisChar = client.read();
          // echo the bytes back to the client:
          server.write(thisChar);
          // echo the bytes to the server as well:
          Serial.write(thisChar);
        }
      }
    }
```

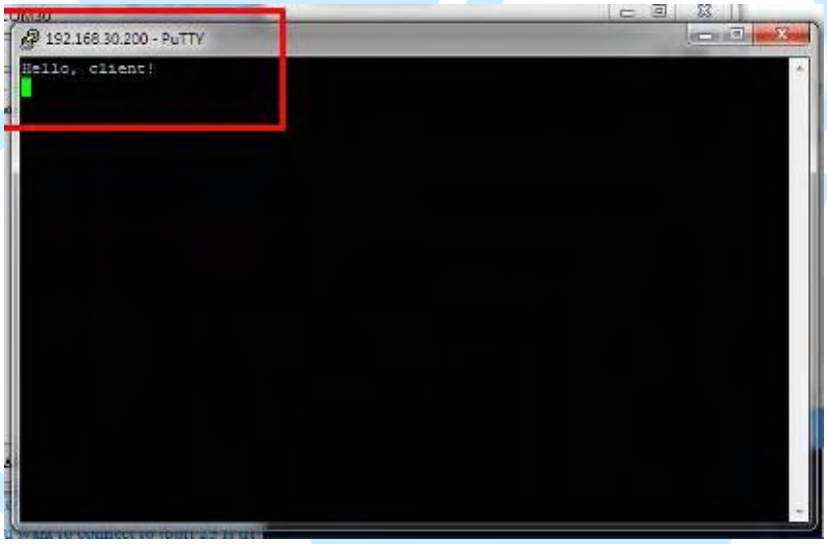
As the picture shown below, we can see Telnet chat room server is ready .

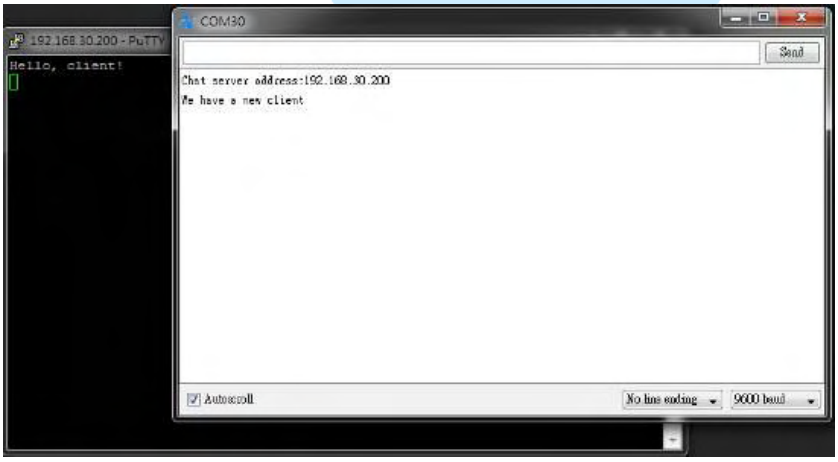


As the picture shown below, we can use the PuTTY program to connect to the Telnet chat room server.



As the picture shown below, we are ready to chat.





8.Telnet Chat Room for Multiple Users

In the previous section, we set up a simple Telnet chat room server for one user. Now we are going to set up a Telnet chat room able to support multiple users.

(a) EduCake

(b) Network Switch

Telnet ChatServer for multiple users

```
#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
byte mac[] = {
  0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);
// the subnet:
IPAddress subnet(255, 255, 255, 0);
```

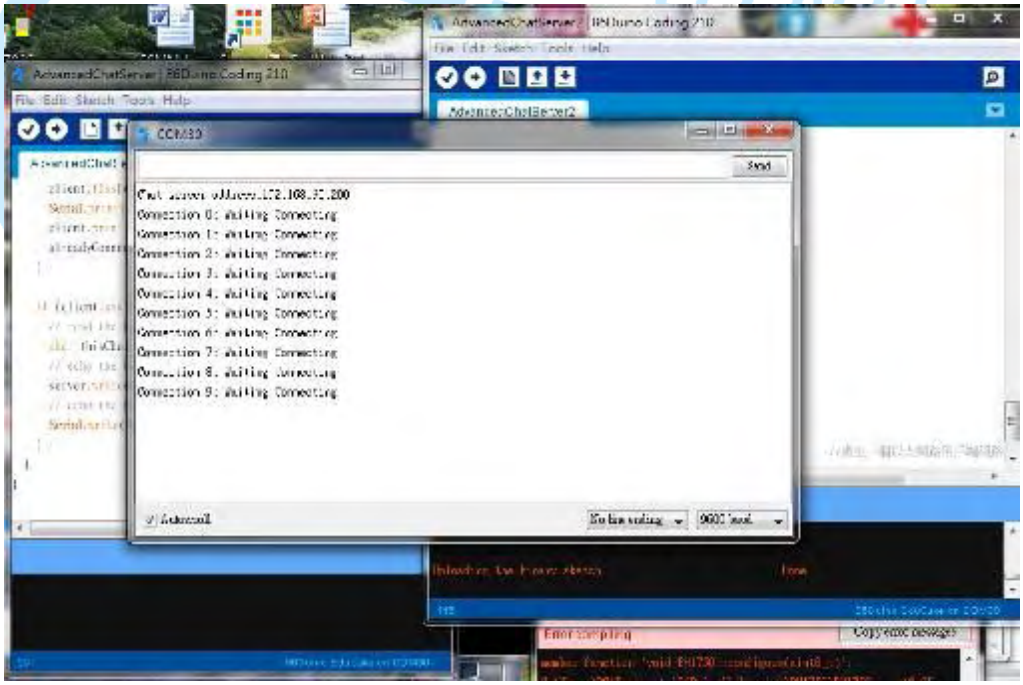
```
// telnet defaults to port 23
EthernetServer server(23);
boolean alreadyConnected ;
boolean ConnectedFlag[10] ;
int connectNo = 0 ;
   EthernetClient client ;
   EthernetClient Connectclient[10];


void setup() {

   // initialize the ethernet device
   Ethernet.begin(mac, ip, gateway, subnet);
   // start listening for clients
   server.begin();
   // Open serial communications and wait for port to open:
   Serial.begin(9600);
      initConnectingFlag() ;

   while (!Serial) {
      ; // wait for serial port to connect. Needed for Leonardo only
   }

   Serial.print("Chat server address:");
   Serial.println(Ethernet.localIP());
   DisplayConnectingStatus() ;
}

void loop() {
   // wait for a new client:
```

```
    client = server.available();

  // when the client sends the first byte, say hello:
  if (client) {
    connectNo = 0 ;
    while (connectNo <10)
    {
        if (!ConnectedFlag[connectNo]) {
          // clead out the input buffer:
              Connectclient[connectNo] =client ;
            Connectclient[connectNo].flush();
          Serial.println("We have a new client");
          client.println("Hello, client!");
          ConnectedFlag[connectNo] = true;
          break ;
        }
       connectNo ++ ;
    }
  }
     connectNo = 0 ;
        while (connectNo <10)
    {
        if (!Connectclient[connectNo].connected())
                ConnectedFlag[connectNo] = false;

        if (Connectclient[connectNo].available() > 0)
        {
          // clead out the input buffer:
          char thisChar = Connectclient[connectNo].read();
          // echo the bytes back to the client:
```

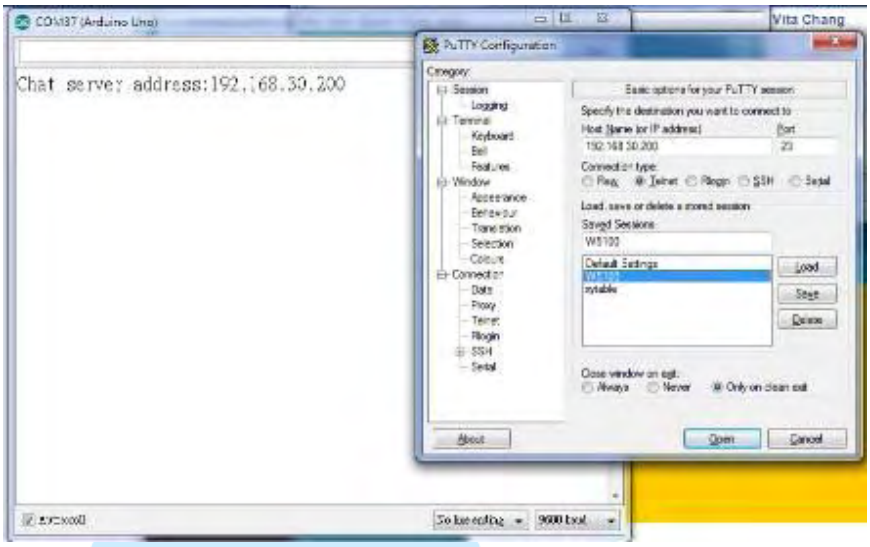
```
                Serial.print("Connect ");
                Serial.print(connectNo);
                Serial.print(":");
                server.write(thisChar);
                // echo the bytes to the server as well:
                Serial.write(thisChar);
            }
            connectNo ++ ;
        }

}

void initConnectingFlag()
{
   for(connectNo=0 ; connectNo < 10 ; connectNo++)
   {
      ConnectedFlag[connectNo] = false ;

   }
}

void DisplayConnectingStatus()
{
   for(connectNo=0 ; connectNo < 10 ; connectNo++)
   {
      if (ConnectedFlag[connectNo])
      {
            Serial.print("Connection ") ;
            Serial.print(connectNo) ;
            Serial.print(": Connected \n") ;
```

```
            }
            else
            {
                  Serial.print("Connection ") ;
                  Serial.print(connectNo) ;
                  Serial.print(": Waiting Connecting \n") ;
            }

          // Connectclient[connectNo]    = server.available();
          }
        }
```

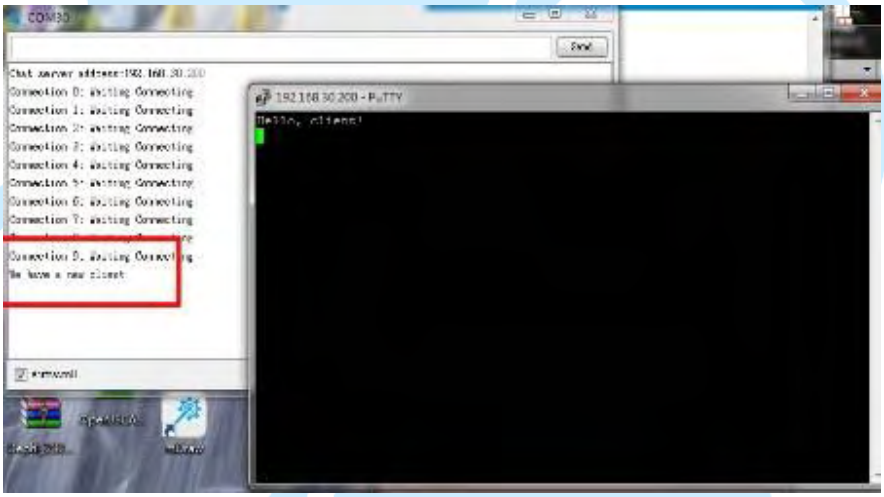
As shown in the picture below, we can see the Telnet chat room server for multiple users is ready .



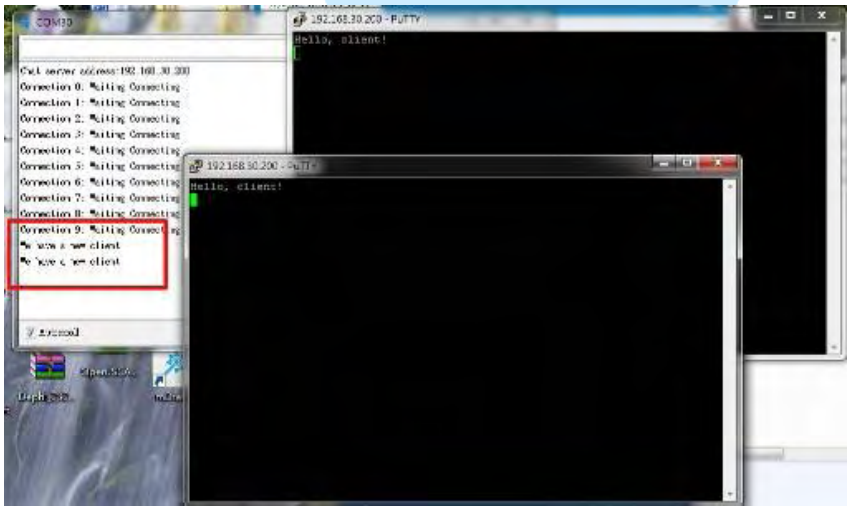
Then, we can use PuTTy to configure the chat server.



As is seen in the picture below, the first user has begun chatting.



As shown below, the second user has begun chatting.



9.Temperature Monitor

If we would like to know the temperature, we will need a thermometer. If we like to know the moisture, we will need hygrometer. If we need to know both temperature and moisture, we might not need both a thermoter and hygrometer. Instead, we can just use a DHT-11 module (as shown in the picture below) to measure temperature and moisture at the same time.

DHT11 Module

Please ensure pin assignment on the module is correct and plug into the Educake breadboard.

DHT11 Pin Assignments

Pin	DTH11	86Duino EduCake
S	Vcc	+5V
2	GND	EduCake GND
3	Signal	EduCake digital pin 7
		

Source：Arduino Programming (37 Sensor Modules)(曹永忠, 許智誠, & 蔡英德, 2015b, 2015f)

We will then use the 86Duino IDE to write a program that allows the EduCake to test for temperature and moisture.

DHT11

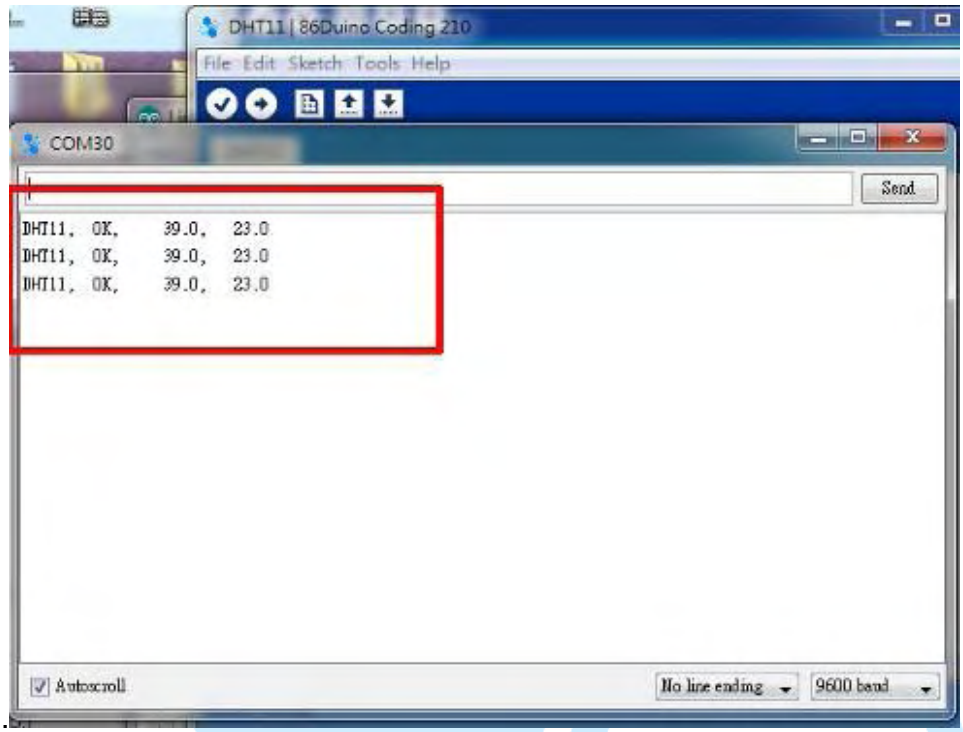
```
int DHpin=7;
byte dat[5];

byte read_data()
{
    byte data;
    for(int i=0; i<8;i++)
    {
        if(digitalRead(DHpin)==LOW)
            {

                while(digitalRead(DHpin)==LOW);
delayMicroseconds(30);
```

```
                    if(digitalRead(DHpin)==HIGH)
                        data |=(1<<(7-i));

                    while(digitalRead(DHpin) == HIGH);
                }
        }
        return data;
}

void start_test()
{
        digitalWrite(DHpin,LOW);
        delay(30);
        digitalWrite(DHpin,HIGH);
        delayMicroseconds(40);
        pinMode(DHpin,INPUT);
      while(digitalRead(DHpin) == HIGH);
          delayMicroseconds(80);
        if(digitalRead(DHpin) == LOW);
            delayMicroseconds(80);

for(int i=0;i<4;i++)
        dat[i] = read_data();

          pinMode(DHpin,OUTPUT);
          digitalWrite(DHpin,HIGH);
    }

void setup()
{
```

```
        Serial.begin(9600);
        pinMode(DHpin,OUTPUT);
    }

    void loop()
    {
        start_test();
        Serial.print("Current humdity = ");
        Serial.print(dat[0], DEC);
        Serial.print('.');
        Serial.print(dat[1],DEC);
        Serial.println('%');
        Serial.print("Current temperature = ");
        Serial.print(dat[2], DEC);
        Serial.print('.');
        Serial.print(dat[3],DEC);
        Serial.println('C');
        delay(700);
     }
```

Resource：(http://shop.dmp.com.tw/INT/products/67)

Below are the testing results


DHT11 | 86Duino Coding 210
File Edit Sketch Tools Help
COM30
Send
DHT11, OK, 39.0, 23.0
DHT11, OK, 39.0, 23.0
DHT11, OK, 39.0, 23.0
Autoscroll
No line ending
9600 baud

10.Temperature and Moisture Monitor Server

In the previous sections, we have learned how to use Telnet and Http prototcol to build a simple web server from the Eudcake. Now with the DHT11 module, we can build a website to retrieve both the temperature and moisture.

As shown in the pictures (a) (b) below, connect one end of Cat 5 cable to the Educake and the other end to the network switch.

(a). EduCake

(b). Network Switch

As with the previous chapters, after installing the 86Duino EduCake development software and opening the IDE, we can write a program shown below to turn the Educake into a web server able to retrieve the temperature and moisture through DHT11 module.

Temperature and Moisture Monitor Server

```
#include <SPI.h>
#include <Ethernet.h>
#include <Wire.h>
#include "dht.h"



#define DHT11_PIN 7
dht DHT;
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
```

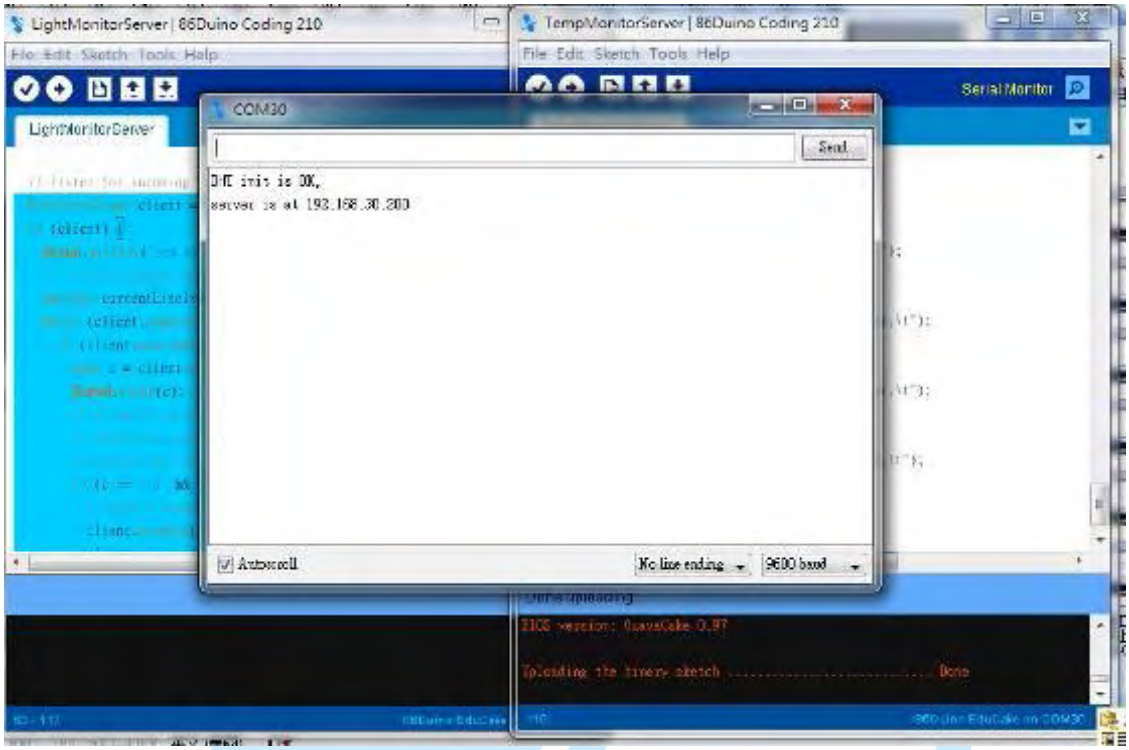
```
byte mac[] = {
  0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
IPAddress ip(192, 168, 30, 200);
IPAddress dnServer(168, 95, 1, 1);
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);
// the subnet:
IPAddress subnet(255, 255, 255, 0);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  int chk = DHT.read11(DHT11_PIN);
   if ( chkDHT(chk) == 0)
        {
             Serial.println("ERROR on init DHT Sensor") ;
            while (true) ;
        }

  // start the Ethernet connection and the server:
```

```
    Ethernet.begin(mac, ip, dnServer, gateway, subnet);
    server.begin();
    Serial.print("server is at ");
    Serial.println(Ethernet.localIP());


}

void loop() {


  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be
closed after completion of the response
          client.println("Refresh: 5");  // refresh the page automatically
every 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
```

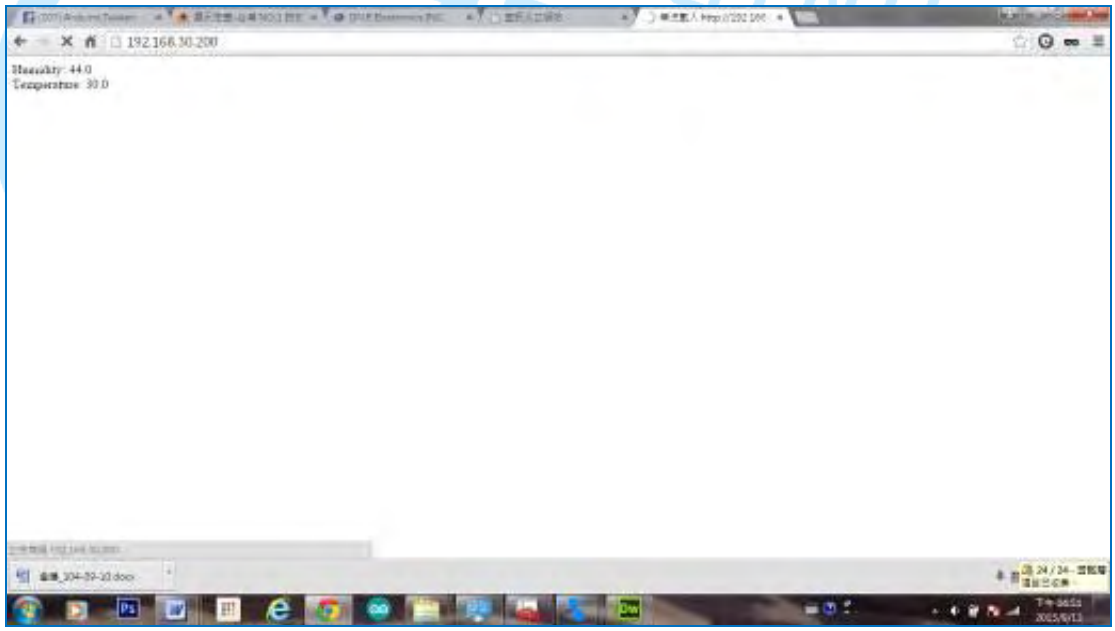
```
                client.println("<html>");
                // output the value of each analog input pin

                  client.print("Humidity: ");
                  client.print(DHT.humidity, 1);
                  client.println("<br>");
                  client.print("Temperature: ");
                  client.print(DHT.temperature, 1);
                  client.println("<br>");

                client.println("</html>");
                break;
              }
              if (c == '\n') {
                // you're starting a new line
                currentLineIsBlank = true;
              }
              else if (c != '\r') {
                // you've gotten a character on the current line
                currentLineIsBlank = false;
              }
            }
          }
          // give the web browser time to receive the data
          delay(1);
          // close the connection:
          client.stop();
          Serial.println("client disconnected");
        }
```

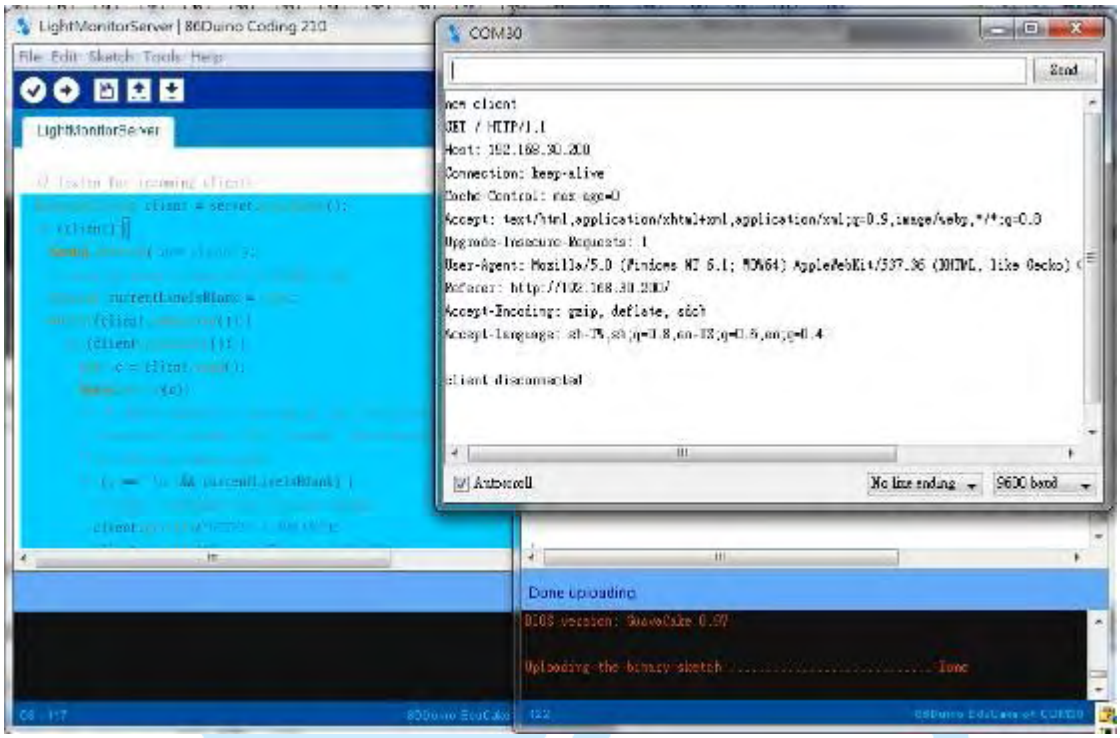
```
    delay(2000);
}

unsigned int chkDHT( int chk )
{
    switch (chk)
  {
    case DHTLIB_OK:
                Serial.println("DHT init is OK,\t");
                return 1;
    case DHTLIB_ERROR_CHECKSUM:
                Serial.println("DHT Checksum error,\t");
                return 0;
    case DHTLIB_ERROR_TIMEOUT:
                Serial.println("DHT Time out error,\t");
               return 0;
    default:
                Serial.println("DHT Unknown error,\t");
               return 0;
  }

}
```

We can see the DHT11 readings from the monitor web server below.



The pictures below show the user of the client can also connect to the monitored web server to see the DHT11 readings.




LightMonitorServer | 86Duino Coding 210
File Edit Sketch Tools Help
LightMonitorServer
COM30
Send
new client
GET / HTTP/1.1
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate, sdch
client disconnected
Autoscroll
No line ending
9600 baud
Done uploading.
Uploading the binary sketch .....done