イーサネット

1.86Duino EduCake ネット紹介

86Duino EduCake 開発ボードは x86 アーキテクチャのオープンソース(Open Source)マイクロコンピュータ学習機であり，内部には高性能の 32 ビット x86 互換性の処理機 Vortex86EX を採用しており， Arduino のプログラミングを互換及び執行可能であり，その特徴としてブレッドボードを内蔵しており，ユーザははんだを行わなくても，すぐに多くの電子部品、測定器及び周辺部品を連結或いは置き換えて電子実験を行うことが可能となっている。それに内蔵された特殊な電気回路保護設計は，誤操作及び I/O ピンの焼けを防ぐことが可能である。これ以外にも，EduCake 開発ボードは外殻は堅固なブレッドボードから構成され，重要な電子部品はすべてこの中に有り、，その周りに常用の I/O インターフェイスが置かれ， EduCak 開発ボードが容易に外部から破壊されず， Arduino 使用に適し 、マイクロコンピュータ及び組み込みシステムの初学者、設計者、趣味愛好家、など如何なる興味を持った人たちに対しても，自分だけの電子装置を作ることが可能となっている。

86Duino EduCake 開発ボードは Arduino 開発ボードとは異なり、W5100 ネット拡張版曹永忠, 許智誠, & 蔡英德, 2015a, 2015c, 2015d, 2015e)を加えることで初めてコンピュータネットワークが稼働し、内蔵のイーサネットモジュールを用いることで、一様に TCP/IP Protocols (TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE, Ethernet)を 86Duino EduCake ハードウェア電気回路上に整合することが可能であり、外部拡張ボードの煩雑さとコストを軽減することが出来る。

86Duino EduCake 開発ボードは Ethernet Library を使用するだけでネットワークに連結し、作動することが可能となっている。

図 186Duino EduCake 開発ボード

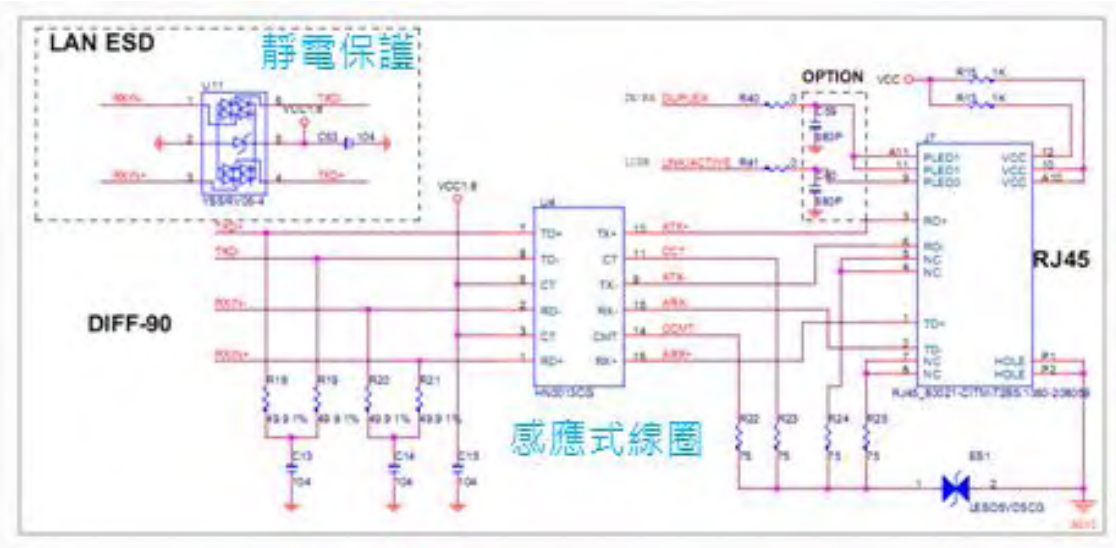
図に示した通り、86Duino EduCake 開発ボードのネットはメインコンピュータ側のイーサネット RJ45 インターフェイスにあり、図に示した通り、私たちは一般的な Cate 5 ネットケーブルを使用し、RJ45 に繋げ、一方を 86Duino EduCake 開発ボード側のネットインターフェイス（下図.(a)の通り)に繋げ、もう一端を下図 b に繋げ、86Duino EduCake 開発ボードネットワーク実体ラインを連結させる。

(a). EduCake ネット接続法

(b). 集線器ネット接続法

図 2 86Duino EduCake 開発ボードのネット接続法を示す

E86Duino EduCake 開発ボード側面は一組のネットインターフェイス(Local Area Network:LAN)を提供し、10/100Mbps 伝送速度をサポートする、ネットインターフェイス(Local Area Network:LAN)上では、下図に示したように、86Duino EduCake 開発ボードに更に静電気保護チップ及び感応式コイルを使用し内部の重要な電子部品を保護している：



86Duino EduCake 開発ボードブート後、下図に示したように、ネットケーブルを LAN インターフェイスに連結し、もし、ネットワークシグナルが正常であれば、ほどなくして LAN インターフェイスの指示ランプが点灯し始め、インターフェイス右側の緑色ランプと、左側のオレンジ色のランプが輝く：

図 386Duino EduCake 開発ボードネットワークランプ表示図

2. 簡単 Web Server

下図に示した通り、私たちは一般的な ate 5 のネットワークケーブルを使用し、RJ45 インターフェイスに接続し、一方を 86Duino EduCake 開発ボード側面のネットワークインターフェイス（下図.(a))に接続し、もう一方をネットワークハブのネットインターフェイス（下図.(b))に繋ぎ、86Duino EduCake 開発ボードネット実体連結回線接続を完成させる。

(a). EduCake ネット接続法　　(b). 集線機ネット接続法

図 486Duino EduCake 開発ボードの接続法指示図

私たちは先の章で述べた方法を順守し、86Duino EduCake 開発ボードの工藤プログラミング設置後、86Duino EduCake 開発ボードの開発キットを開く：Sketch IDE が開発ソフトウェアを制御し、下図に示したような WebServer テストプロ部ラミングを読み取る。私たちは 86Duino EduCake 開発ボードを簡易的なネットサーバーと変更し、並びに Analog Port0 ~Port の状態をプレビューに表示する。

表 1 WebServer テストプログラミング

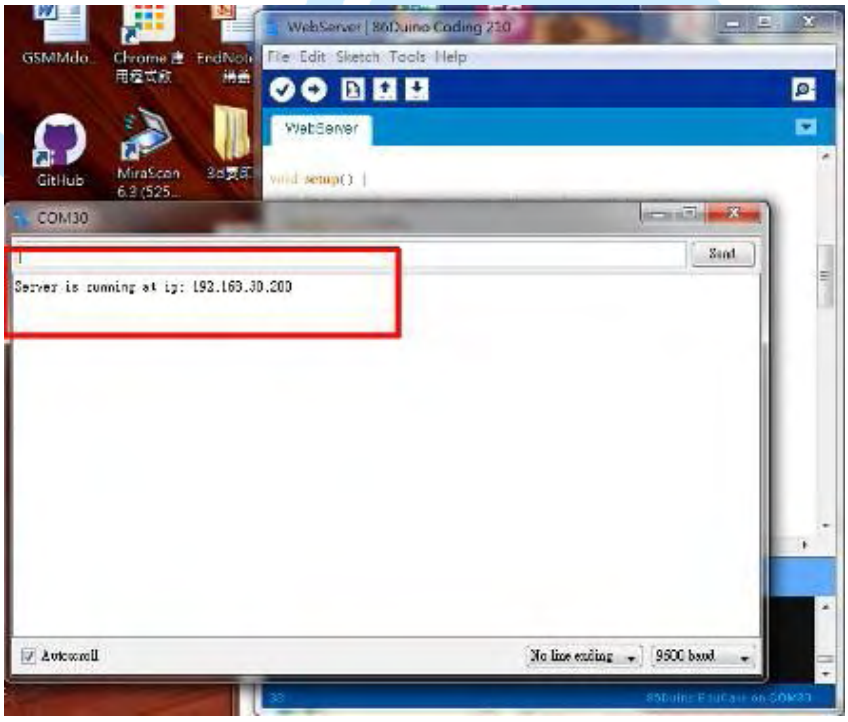
```
#include <SPI.h>
#include <Ethernet.h>          //   ネットが必ず必要なものを使用

// 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
};      //この MAC ADRESS をこの educake の MAC ADDRESS として使用
IPAddress ip(192, 168, 30, 200);          //予め設置したネット IP アドレスを，読者は自由に変更することが可能
IPAddress dnServer(168, 95, 1, 1);       /予め設置した DNS サーバを，本書では Hinet の DNS サーバとし，読者は自由に変更可能
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);       //予め設置したゲートウェイのアドレス(つまり Rounter 或いは AP のアドレス)を，作者がコーディングした環境ゲートウェイのアドレスとし，読者は自由に変更可能
// the subnet:
IPAddress subnet(255, 255, 255, 0);          //サブネット遮断，本書では Class C とする

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);                  //サーバ宣告並びに Port 80 を通信ポートとして使用

void setup() {
   // Open serial communications and wait for port to open:
   Serial.begin(9600);                  //監視画面の通信速度宣告
   while (!Serial) {
      ; // wait for serial port to connect. Needed for Leonardo only
   }

   // start the Ethernet connection and the server:
   Ethernet.begin(mac, ip, dnServer, gateway, subnet);       //上述したネット設定を使用 WEB サーバを起動すす
   server.begin();                  // Web サーバ起動
   Serial.print("Server is running at ip: ");                  //サーバデータ出力
   Serial.println(Ethernet.localIP());                  //  サーバ IP アドレス出力
}

void loop() {
   // listen for incoming clients
   EthernetClient client = server.available(); //もし WEB サーバに入るのなら，
```

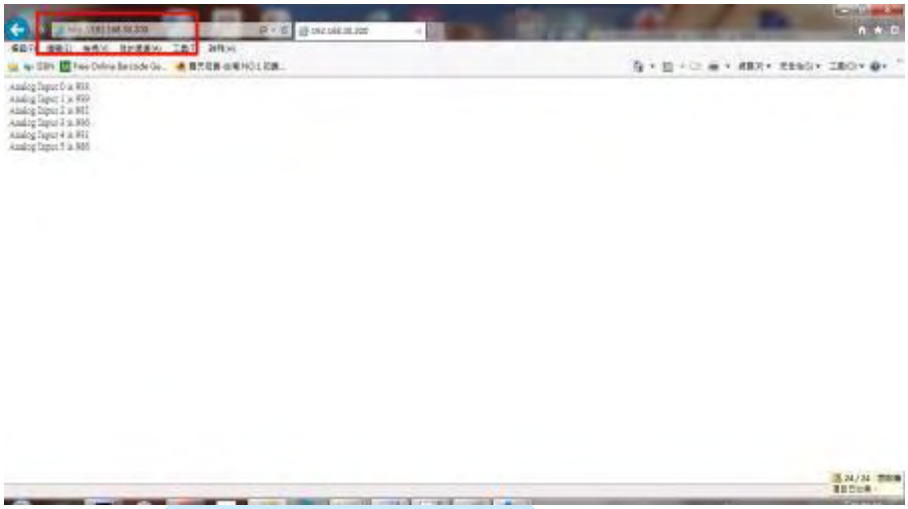
```
Client 連線端を起動
   if (client) {         / Client 連接端起動成功
      Serial.println("New Client");        // "新連線"出力
      // an http request ends with a blank line
      boolean currentLineIsBlank = true;                    //
      while (client.connected()) {          //連線成功
         if (client.available()) {             //連線の Client データ転送
            char c = client.read();                 //連線の Client データ変数読み取り
            Serial.write(c);                    /変数出力
            // 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) {         //変数 c を置き換え目下のデ
ータを空にする
               // send a standard http response header
               client.println("HTTP/1.1 200 OK");         / Http  データ送信(必須)
               client.println("Content-Type: text/html");         // Http データ送信(必須)
client.println("Connection: close");   // the connection will be closed after completion
of the response              // Http データ送信(必須)
               client.println("Refresh: 5");   // refresh the page automatically every 5 sec
            // Http データ送信(必須)
               client.println();          // Http データ送信(必須)
               client.println("<!DOCTYPE HTML>");          // Http データ送信(必須)
               client.println("<html>");          // Http データ送信(必須)
               // output the value of each analog input pin
               for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
     //アナログピンを 0 から 5 としデータ読み取り
                  int sensorReading = analogRead(analogChannel);
                  client.print("Analog Input ");              //画面データ送信
                  client.print(analogChannel);         //アナログピン通信ポート送信
            client.print(" is ");               //画面データ送信
                  client.print(sensorReading);         //アナログピン通信ポートが読み
取ったデータ送信
                  client.println("<br />");         //ネット変換キー出力
               }
               client.println("</html>");         //ネット Tag データ送信(必須)
               break;
            }
```

```
            if (c == '\n') {         /新な一行
              // you're starting a new line
              currentLineIsBlank = true;
            }
            else if (c != '\r') {          //変換キー
              // you've gotten a character on the current line
              currentLineIsBlank = false;
            }
          }
        }
        // give the web browser time to receive the data
        delay(1);        //一秒のズレ
        // close the connection:
        client.stop();          //連線の Client 連線端スイッチ(切り替えを行わないと，抜ける)
        Serial.println("Client Disconnected");          //断線出力
    }
}
```

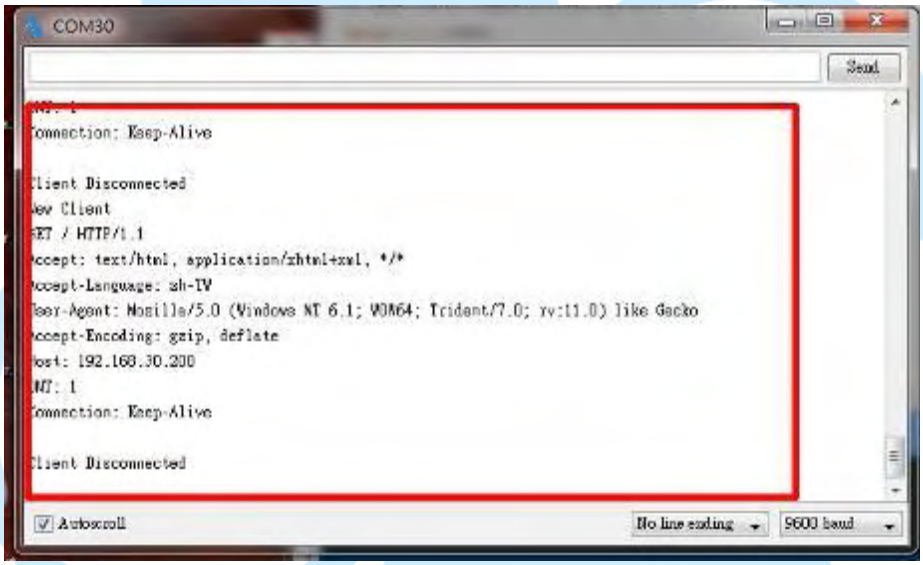
下図に示した通り、読者は本実験—WebServer テストプログラミング結果画面を見ることが可能である。



(a). WebServer テストプログラミング



(b). ウェブブラウザ連結起動



(c). WebServer 連線後の結果画面

図 5WebServer テストプログラミング結果画面

3.DHCP を使用し Web Server 組み立てる

下図に示した通り、私たちは一般的な ate 5 のネットワークケーブルを使用し、RJ45 インターフェイスに接続し、一方を 86Duino EduCake 開発ボード側面のネットワークインターフェイス（下図.(a))に接続し、もう一方をネットワークハブのネットインターフェイス（下図.(b))に繋ぎ、86Duino EduCake 開発ボードネット実体連結回線接続を完成させる。

(a). EduCake ネット接続法

(b). 集線機ネット接続法

図 686Duino EduCake 開発ボードのネット接続法は図の通り

我先の章で述べたやり方に従い、86Duino EduCake 開発ボードのドライバー組込み後、86Duino EduCake 開発ボードの開発キットを開こう：Sketch IDE は開発ソフトを整合し、プログラミング後、下記に記した WebServer テストプログラミングを表示するので、86Duino EduCake 開発ボードを簡易的な WebServe とし、86Duino EduCake 開発ボードは WebServer が取得した IP アドレスをカスタマイズしたものではなくなり、 DHCP サーバが取得した IP アドレスが、Analog Port0 ~Port5 の状態で画面上に表示される。

表 2 WebServer テストプログラミング一

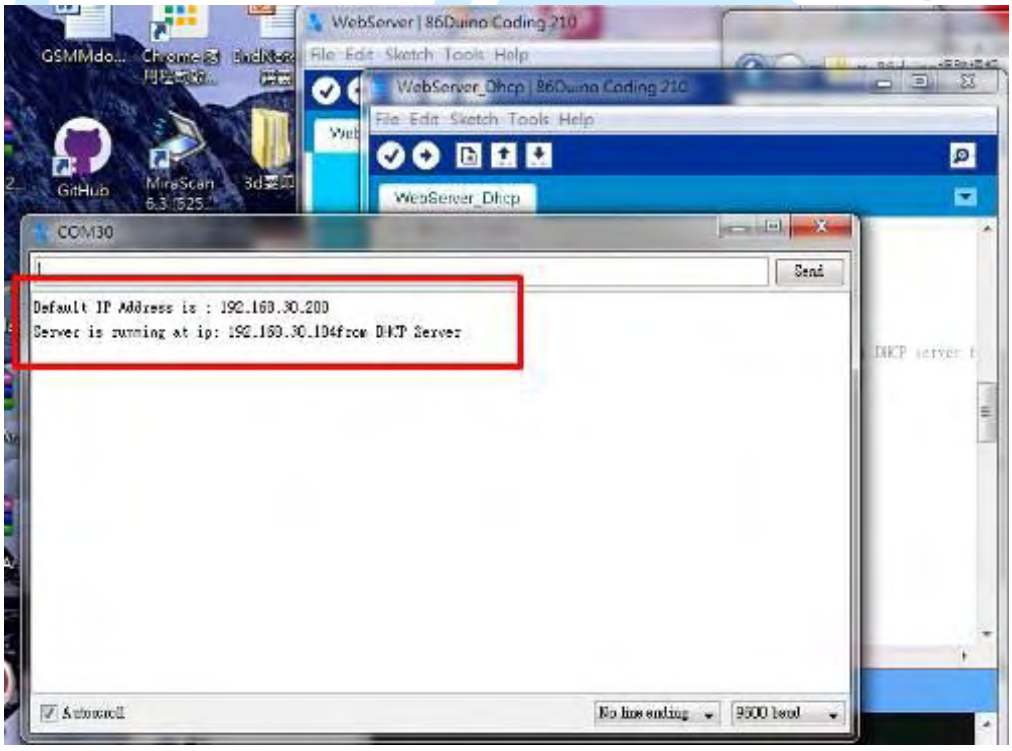
```
#include <SPI.h>
#include <Ethernet.h>        //   ネットが必要とするものを使用

// 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
};          //この  MAC ADRESS をこの educake  の MAC ADDRESS
IPAddress ip(192, 168, 30, 200);     として使用//あらかじめ設定した IP アドレスを，読者は自ら使用するものに改めることが出来る

IPAddress dnServer(168, 95, 1, 1);        //あらかじめ設定した DNS サーバを，本書では Hinet の DNS サーバとし，読者は自身が使用する DNS サーバに変更可能
// the router's gateway address:
      IPAddress gateway(192, 168, 30, 254); //あらかじめ設定したゲートウェイのアドレス（つまり Rounter 或いは AP のアドレス）を、本書では作者がコーディングした環境のゲートウェイアドレスとし、読者は自らそれを変更し用いることが可能である
// the subnet:
IPAddress subnet(255, 255, 255, 0);            //サブネットを遮断し，本書では Class C とする

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);       //サーバ宣告並びに Port 80 通信ポート使用

void setup() {
   // Open serial communications and wait for port to open:
   Serial.begin(9600);        //画面通信速度率宣告
      Serial.println("Now Program Start") ;

   while (!Serial) {
```

```
      ; // wait for serial port to connect. Needed for Leonardo only
   }

   // アクティブ化した Ethernet を連線，予め設定した DHCP が IP アドレス取得
   if (Ethernet.begin(mac) == 0) {
      Serial.println("I can't get any   IP address from DHCP Server");
      // IP 取得したアドレスは，使用不可
      for(;;)
         ;
   }
   // IP アドレス出力
   Serial.print("Default IP Address is : "); //サーバデータ出力
   Serial.println(ip);            //サーバ IP アドレス出力

//     Ethernet.begin(mac, ip, dnServer, gateway, subnet);
     Ethernet.begin(mac);      // もし DHCP サーバが ip を取得して用いるなら,必ずこのコーディングが必要

   server.begin();        //アクティブ化 Web サーバ運用
   Serial.print("Server is running at ip: ");       //サーバデータ出力
   Serial.print(Ethernet.localIP());          //サーバデータ出力
   Serial.print("from DHCP Server \n");}         //サーバデータ出力


void loop() {
   // listen for incoming clients
   EthernetClient client = server.available();         //もし WEB サーバに入ったとしたら、Client 連線端アクティブ化
   if (client) {        //もし Client 連先端が成功したらアクティブ化
      Serial.println("New Client");             // "新連線"出力
      // an http request ends with a blank line
      boolean currentLineIsBlank = true;
      while (client.connected()) {       //連線成功したら
         if (client.available()) {           //連線の Client にデータ転送したら
            char c = client.read();        //連線の Clien にデータ出力したら
Serial.write(c);           //変数 c 出力
            // if you've gotten to the end of the line (received a newline
```

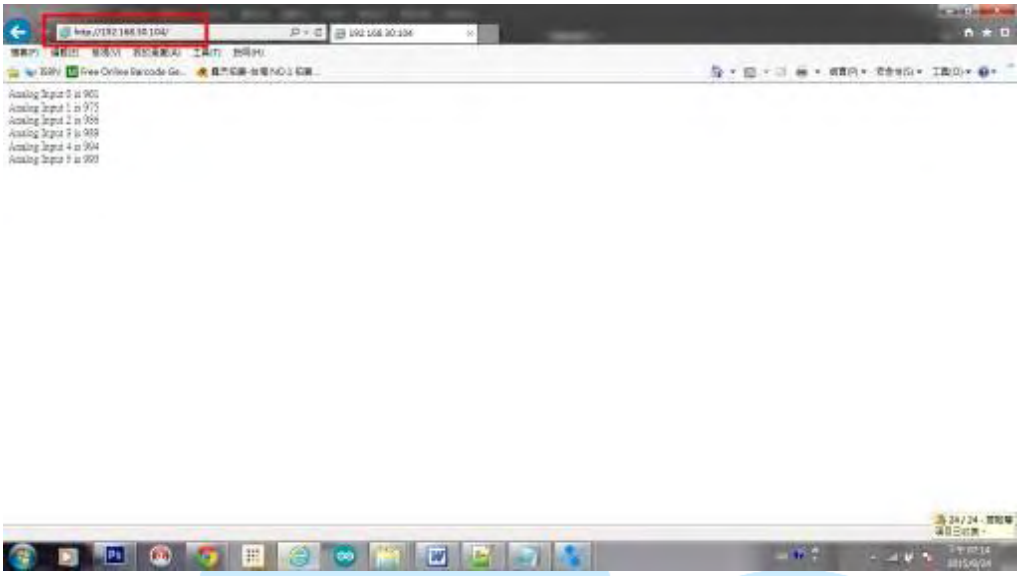
```
            // character) and the line is blank, the http request has ended,
            // so you can send a reply
            if (c == '\n' && currentLineIsBlank) {       //変数 c は目前のデータを空
にする
              // send a standard http response header
              client.println("HTTP/1.1 200 OK");       // Http  表題データ送信  (必
須)
              client.println("Content-Type: text/html");           // Http 表題データ送
信(必須)
              client.println("Connection: close");   // the connection will be closed
after completion of the response      // Http 表題データ送信(必須)

              client.println("Refresh: 5");   // refresh the page automatically every
5 sec     // Http 表題データ送信(必須)

              client.println();        // Http 表題データ送信(必須)

              client.println("<!DOCTYPE HTML>");       // Http 表題データ送信
(必須)
              client.println("<html>");        // Http 表題データ送信(必須)

              // output the value of each analog input pin
              for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
    //アナログピン 0 から 5 データ読み取り
                int sensorReading = analogRead(analogChannel);
                client.print("Analog Input ");            //画面データ送信
                client.print(analogChannel);            //いくつかのアナログ通信ポ
ート送信
                client.print(" is ");        //画面データ送信
                client.print(sensorReading);            //アナログデータ通信ポート
データ読み取り後送信
                client.println("<br />");        //ネット改行キー送信
              }
              client.println("</html>");        //ネット Tag データ送信(必須)
              break;
            }
            if (c == '\n') {        //新たな一行
              // you're starting a new line
```

```
            currentLineIsBlank = true;
          }
          else if (c != '\r') {          //もし改行キーでなくば
            // you've gotten a character on the current line
            currentLineIsBlank = false;
          }
        }
      }
      // give the web browser time to receive the data
      delay(1);       //一秒遅延
      // close the connection:
          client.stop();          //連線の Client 連先端開閉(必須)

      Serial.println("Client Disconnected");          //連線断線送信
  }
}
```

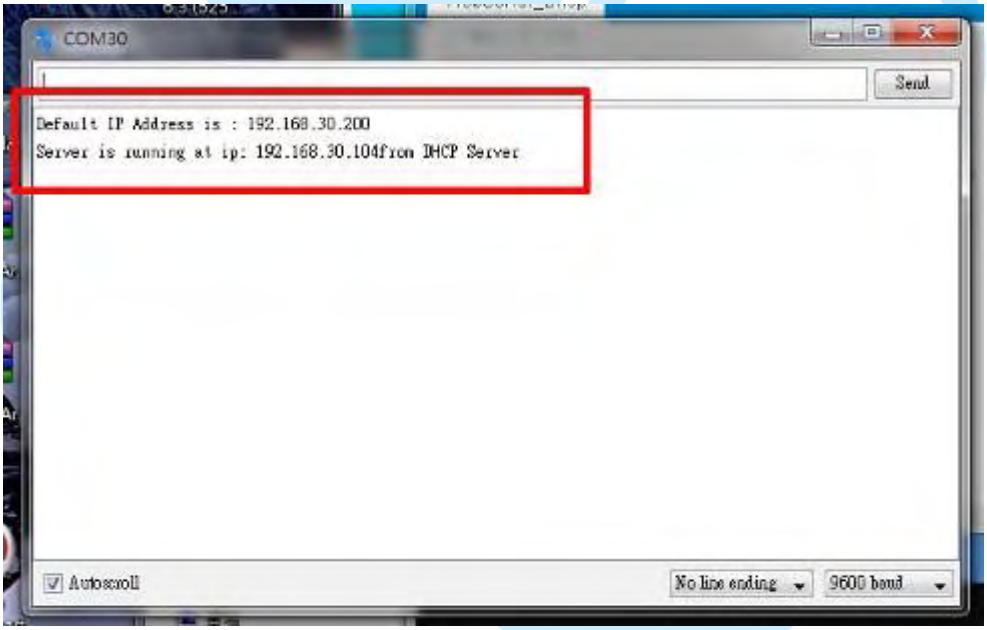
下図で示した通り、読者は本実験により WebServer テストプログラミングの結果画面を見ることが可能である。



(a). WebServer テストプログラミング開始画面



(b). ウェブブラウザ連線アクティブ



(c). WebServer 連戦後結果画面

図 7 WebServer プログラミング結果画面

4.Telnet クライアントプログラミング

まず、下図に示したとおり、一般的な Cate 5 ネット回線を使用し、RJ45 を加え、86Duino EduCake を接続し、開発ボード側のネットインターフェイス(下図.(a))に接続し、ネット集線機のネットインターフェイス(下図.(b))にもう一方を接続し、86Duino EduCake 開発ボードの実態回路を完成させる。

(a). EduCake ネット接続法　　(b). 集線機ネット接続法

図 8 86Duino EduCake 開発ボードのネット接続法は下図の通り

先の章で述べたやり方にしたがい、86Duino EduCake 開発ボードのドライバーを組み込んだのち、86Duino EduCake 開発キットを開こう：Sketch IDE は開発ソフトを整合し、コーディング後、下図に示したような Telnet ユーザプログラミングテストプログラミングを表示するので、86Duino EduCake 開発ボードを簡易的な Telnet ユーザ用の仕事場とする。

表 3 Telnet ユーザテストプログラミング

```
#include <SPI.h>
#include <Ethernet.h>          //   ネット必須のものを使用

// 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
};          //この MAC ADRESS  をこの educake  の MAC ADDRESS とみなし使用
IPAddress ip(192, 168, 30, 200);          //あらかじめ設定したネットの IP アドレスを使用し，読者は自由に変更可能
IPAddress dnServer(168, 95, 1, 1);       //予め設定した DNS サーバを，本書では Hinet の DNS サーバとし，読者は自由に使用可能
// the router's gateway address:
      IPAddress gateway(192, 168, 30, 254);       /予め設定したゲートのアドレス（つまり Rounter 或いは AP のアドレス）を、本書は作者がコーディングした環境のゲートアドレスとし、読者は自由に変更使用が可能である
// the subnet:
IPAddress subnet(255, 255, 255, 0);          //サブネット遮断，本書は Class C とする


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

// 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;          // TCP/IP の  Client 宣告

void setup() {
   // start the Ethernet connection:
   Ethernet.begin(mac, ip, dnServer, gateway, subnet);       // Ethernet  連線アクティブ

   // Open serial communications and wait for port to open:
```

```
  Serial.begin(9600);           //画面通信速度宣告
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  // give the Ethernet shield a second to initialize:
  delay(1000);              // 1000 秒遲延
  Serial.println("connecting...");          //連線中字句....出力

  // if you get a connection, report back via serial:
  if (client.connect(server, 23)) {         //もし Client が server(140, 112, 172, 11)につながったとしたら; PTT ネットは静甲
    Serial.println("connected");          //連線成功字句....出力
  }
  else {
    // if you didn't get a connection to the server:
    Serial.println("connection failed");       //連線失敗字句....出力
  }
}

void loop()
{
  // if there are incoming bytes available
  // from the server, read them and print them:
  if (client.available()) {             //もし Clien がポストバックしたら
    char c = client.read();             // Client 読み取り，変数 c 保存
    Serial.print(c);       //変数 c 出力  //
  }

  // 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) {          /画面上から文字を入力したとしたら，
    char inChar = Serial.read();          //画面文字読み取り，並びに変数 c 入力
    if (client.connected()) {         //連線中だとしたら
      client.print(inChar);           //入力文字 Client に転送
    }
  }
```

```
  // if the server's disconnected, stop the client:
  if (!client.connected()) {          //もし Client が既に断線していたら
    Serial.println();
    Serial.println("disconnecting.");          ////既に断線出力
    client.stop();                //Client 開閉
    // do nothing:
    while (true);          //永久ループ，もう二度としない
  }
}
```

下図に示した通り、読者は本実験 Telnet ユーザプログラミングテスト画面を見ることが出来る。

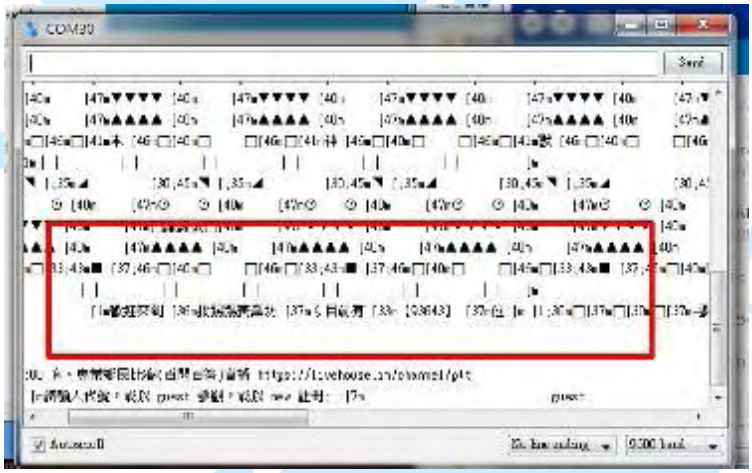


図 9 Telnet ユーザプログラミング結果画面

5.文字型 Browser 用戸端程式

下図に示した通り、私たちは一般的な ate 5 のネットワークケーブルを使用し、RJ45 インターフェイスに接続し、一方を 86Duino EduCake 開発ボード側面のネットワークインターフェイス（下図.(a))に接続し、もう一方をネットワークハブのネットインターフェイス（下図.(b))に繋ぎ、86Duino EduCake 開発ボードネット実体連結回線接続を完成させる。

(a). EduCake 網路接法　　(b). 集線器網路接法

図 1 86Duino EduCake 開発ボードネット接続法示す

先に挙げた方法で、86Duino EduCake 開発ボードドライバープログラミング組込み後、86Duino EduCake 開発ボードの開発キットを開く：Sketch IDE が開発プログラミングを整合し、コーディング後、下に示した Telnet ユーザテストプログラミングを示し、86Duino EduCake 開発ボードを簡易的なウェブブラウザ文字処理機に変化させることで、文字フィルターの方式により、理想的なデータを探すことが可能となる。

表 4 文字型 Browser ユーザプログラミング(WebClient)

```
#include <SPI.h>
#include <Ethernet.h>          //   ネット必須のものを使用


// 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
};          //この MAC ADRESS をこの educake の MAC ADDRESS
IPAddress ip(192, 168, 30, 200);として使用            //予め設定したネット IP アドレスを，読者は自由に変更使用可能
IPAddress dnServer(168, 95, 1, 1);      //予め設定した DNS サーバを、本書では Hinet の DNS サーバとして、読者は自由に使用可能
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);       //予め設定したゲートアドレス（つまり Rounter 或いは AP のアドレス）を本書では作者がコーディングした環境のゲートのアドレスとし、読者は自由に使用可能
// the subnet:
IPAddress subnet(255, 255, 255, 0);           //サブネット遮断，本書では Class C とする

// 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 のアドレス
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);          //画面の通信速度宣告
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {        // DHCP サーバ ip アドレス等データ発信要求
    Serial.println("Failed to configure Ethernet using DHCP");        // DHCP サー
```

```
バ ip アドレス発信失敗、警告データ発信
      // 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);       //システムがあらかじめ設定した ip アドレス使用


   }
   // give the Ethernet shield a second to initialize:
   delay(1000);                  // 1000 秒遅延
   Serial.println("connecting...");          //連線中字句....出力



   // if you get a connection, report back via serial:
   if (client.connect(server, 80)) {         //もし lient が server(74,125,232,128) www.google.com のアドレス接続に成功したら
      Serial.println("connected");          //連線成功字句....出力
      // Make a HTTP request:
      //以下は固定の http  通信  protocol
      client.println("GET /search?q=arduino HTTP/1.1");
      client.println("Host: www.google.com");            //以下  www.google.com への接続告知
      client.println("Connection: close");              //接続後開閉
      client.println();
   }
   else {
      // if you didn't get a connection to the server:
      Serial.println("connection failed");       // www.google.com 接続失敗，使用者失敗宣告
   }
}

void loop()
{
   // if there are incoming bytes available
   // from the server, read them and print them:
   if (client.available()) {              //もし Client ポストバックが有ったとしたら
      char c = client.read();           // Client 読み取り ，並びに変数 c
      Serial.print(c);              //変数 c 出力
```

```
    }

    // if the server's disconnected, stop the client:
    if (!client.connected()) {          //もし lient 既に連線開閉していたら
      Serial.println();
      Serial.println("disconnecting.");                  //連線成功字句中断....出力
      client.stop();          //連現中止

      // do nothing forevermore:
      while (true);                 //永久ループ中断プログラム
    }
}
```

下図に示した通り、読者は本実験で文字型 Browser ユーザプログラミング結果画面を見ることが出来る。

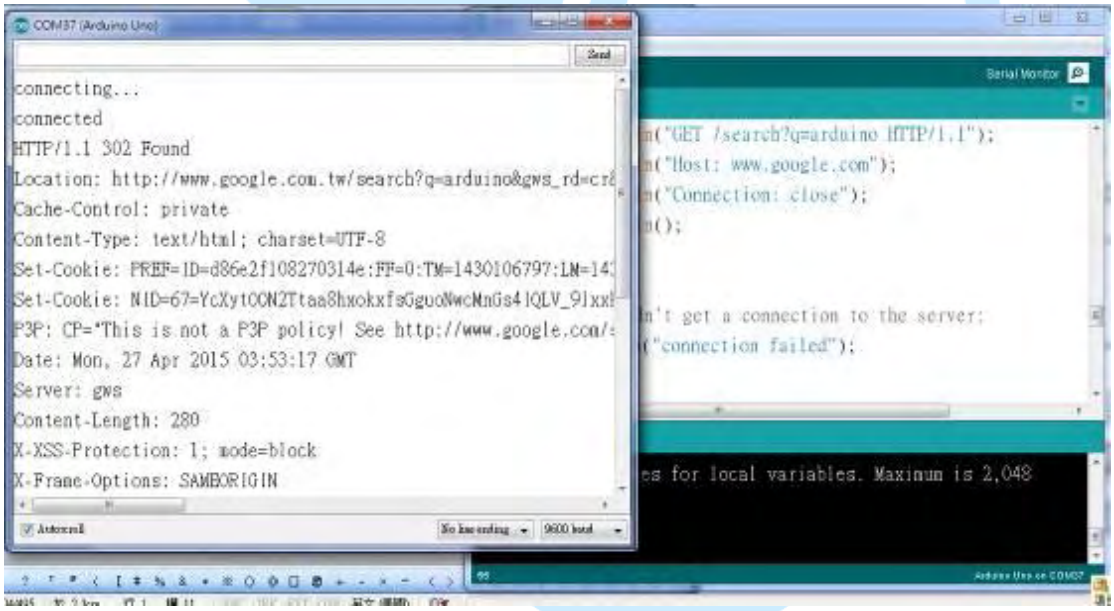


図 11 文字型 Browser ユーザプログラミング画面

6. ネットタイマデータ取得

下図に示した通り、一般的な Cate 5 ネットケーブルを使用し、RJ45 コネクタに組合せる。一端を 86Duino EduCake 開発ボード側面のネットワークインターフェイス（下図.(a)に示した通り)に繋ぎ、一端をネット集線機ネットワークインターフェイス（下図.(b))に繋ぎ、86Duino EduCake 開発ボードネットワーク実体回線連線を完成させる。

(a). EduCake ネットワーク接続法

(b). 集線機ネットワーク接続法

圖 1286Duino EduCake 開発ボードのネット接続法示す

先に述べたことを順守し、86Duino EduCake 開発ボードにドライバープログラミング組込み後、86Duino EduCake 開発ボードの開発キットを開こう：Sketch IDE が開発プログラミングを整合し、下図に示した通りのネットワークの流れのテストプログラミングをコーディングしよう。この取得には時間がかかる。

表 1 ネットワーク測定プログラミング(UdpNtpClient)

```
#include <SPI.h>
#include <Ethernet.h>          //   ネットワーク必須のものを使用#include <Ether-
netUdp.h>      //   ネットワーク UDP 通信協定必須のものを使用

// 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
```

```
};          //この MAC ADRESS をこの educake の MAC ADDRESS と見なして使用
IPAddress ip(192, 168, 30, 200);          //あらかじめ設定したネットワーク IP アドラスを、読者は自由に仕様変更可能
IPAddress dnServer(168, 95, 1, 1);       //あらかじめ設定した DNS サーバを、本書は Hinet の DNS サーバとし、読者は自由に変更使用可能
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);       //あらかじめ設定したゲートのアドレス（つまり Rounter 或いは AP のアドレス）を、本書では作者がコーディングした環境ゲートウェイのアドレスとし、読者は自由に変更使用可能
// the subnet:
IPAddress subnet(255, 255, 255, 0);          //サブネットワーク遮断、本書は Class C とする

unsigned int localPort = 8888;          /必須の設定 local port を UDP に使用
          // local port to listen for UDP packets

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

const int NTP_PACKET_SIZE = 48; // NTP time stamp is in the first 48 bytes of the message  大小 48 ビットをパケットする

byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and outgoing packets   パケットデータ変数受け入れ

// A UDP instance to let us send and receive packets over UDP
EthernetUDP Udp;        / UDP ネットワーク宣告

void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(9600);          //画面通信速度率宣告
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  // start Ethernet and UDP
  if (Ethernet.begin(mac) == 0) {        //DHCP サーバ ip アドレス等データ発信要
```

```
求
      Serial.println("Failed to configure Ethernet using DHCP");        //DHCP サーバ
ip アドレス失敗発信、警告データ発信
      // no point in carrying on, so do nothing forevermore:
   Ethernet.begin(mac, ip, dnServer, gateway, subnet);       //システムがあらかじめ
設定した ip アドレス使用
   }
   Udp.begin(localPort);        // UDP  起動
}

void loop()
{
   sendNTPpacket(timeServer); // send an NTP packet to a time server      time.nist.gov
NTP server をパケットしてサーバ伝送

   // wait to see if a reply is available
   delay(1000);              // 1000 秒遅延
   if ( Udp.parsePacket() ) {          //もし  time.nist.gov NTP server  サーバパケット
伝送としたら
      // We've received a packet, read the data from it
      Udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the buffer
            //パケット読み取り

      //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]);             //
高ビットを highWord  変数へ伝送
      unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);       //低ビ
ットを lowWord 変数へ転送
      // 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;                    //高ビ
ットで変数形成
      Serial.print("Seconds since Jan 1 1900 = " );              //時間出力
      Serial.println(secsSince1900);
      // now convert NTP time into everyday time:
      Serial.print("Unix time = ");          //印出 Unix time =
```

```
      // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
      const unsigned long seventyYears = 2208988800UL;          // uni データ初期時
間設定
      // subtract seventy years:
      unsigned long epoch = secsSince1900 - seventyYears;       //取得時間差計算
      // print Unix time:
      Serial.println(epoch);            //時間差出力


      // print the hour, minute and second:
      Serial.print("The UTC time is ");            // UTC is the time at Greenwich Merid-
ian (GMT)
      Serial.print((epoch    % 86400L) / 3600); // print the hour (86400 equals secs per
day)  何日間共有か出力
      Serial.print(':');
      if ( ((epoch % 3600) / 60) < 10 ) {
        // In the first 10 minutes of each hour, we'll want a leading '0'    何時間か出力
        Serial.print('0');
      }
      Serial.print((epoch    % 3600) / 60); // print the minute (3600 equals secs per mi-
nute)  何分か出力
      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      何秒か出力
    }
    // wait ten seconds before asking for the time again
    delay(10000);                  // 10000 秒遅延
}

// NTP パケット関数送信
// send an NTP request to the time server at the given address
unsigned long sendNTPpacket(char* address)      // NTP パケット関数送信
{
              // NTP パケット関数送信
    // set all bytes in the buffer to 0
```

```
  memset(packetBuffer, 0, NTP_PACKET_SIZE);     // packetBuffer 設定
  // 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              Port 123 を
利用し address 変数中のアドレスに送信
  Udp.write(packetBuffer, NTP_PACKET_SIZE);          // packetBuffer 変数デー
タ送信
  Udp.endPacket();              //終了送信
}
```

下図に示した通り、読者は本実験によりネットワークテストプログラミング結果を画面で見ることが可能である。

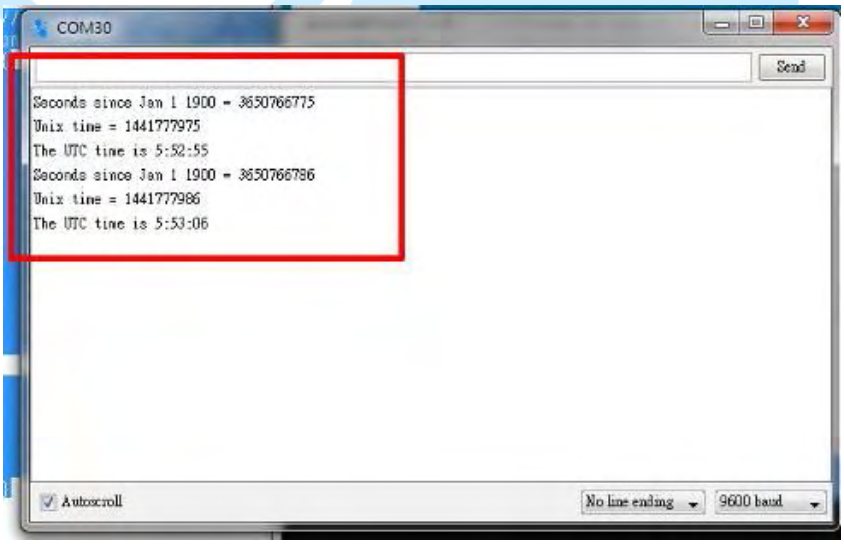


図 13 ネットワークプログラミングテスト結果画面

7.Telnet 簡単チャットルーム

下図に示した通り、一般的な Cate 5 ネットケーブルを使用し、RJ45 コネクタに組合せる。一端を 86Duino EduCake 開発ボード側面のネットワークインターフェイス（下図.(a)に示した通り)に繋ぎ、一端をネット集線機ネットワークインターフェイス（下図.(b))に繋ぎ、86Duino EduCake 開発ボードネットワーク実体回線連線を完成させる。

(a). EduCake ネットワーク接続法　　(b). 集線器ネットワーク接続法

図 14 86Duino EduCake　開発ボードのネットワーク接続法指示図

先に挙げた方法で、86Duino EduCake 開発ボードドライバープログラミング組込み後、86Duino EduCake 開発ボードの開発キットを開く：Sketch IDE が開発プログラミングを整合し、下図に示した Telnet チャットルームのテストプログラミングをコーディングし、86Duino EduCake 開発ボードに簡易的 Telnet チャットルームを作動させることが可能となる。

表 6 Telnet Telnet 簡単チャットルームテストプログラミング(ChatServer)

```
#include <SPI.h>
#include <Ethernet.h>          //   ネット必須のものを使用
```

```
// 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
};          //この MAC ADRESS はこの educake  の MAC ADDRESS を使用
IPAddress ip(192, 168, 30, 200);          //あらかじめ設定した IP アドレスを、読者は自由に変更使用可能
IPAddress dnServer(168, 95, 1, 1);       //あらかじめ設定した DNS サーバーを，本書は Hinet の DNS サーバーとし，読者は自由に変更使用可能
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);       //あらかじめ設定したゲートのアドレス（つまり Rounter 或いは AP のアドレス）を、本書は作者がコーディングしたゲートのアドレスとし、読者は自由に変更使用可能
// the subnet:
IPAddress subnet(255, 255, 255, 0);           //サブネット遮断、本書は Class とする

// telnet defaults to port 23
EthernetServer server(23);           //
 boolean alreadyConnected = false; // whether or not the client was connected previously 連線フラグ

void setup() {
  // initialize the ethernet device
  Ethernet.begin(mac, ip, gateway, subnet); //イーサネット起動
  // start listening for clients
  server.begin();           // Port23 のサーバ起動
  // Open serial communications and wait for port to open:
  Serial.begin(9600);           //画面の通信速度宣告
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  Serial.print("Chat server address:");          // Chat server address:出力
  Serial.println(Ethernet.localIP());           //アドレス出力
}

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

```
  EthernetClient client = server.available(); //ネットワークユーザ端生産

  // when the client sends the first byte, say hello:
  if (client) {        //ネットワークアクセス
    if (!alreadyConnected) {//もし新しい連線だとしたら
      // clead out the input buffer:
      client.flush();           //バッファーデータ更新
      Serial.println("We have a new client");          //新連線出力
      client.println("Hello, client!");                // Hello 歓迎の言葉送信
      alreadyConnected = true;          //連線設定済み

    }

    if (client.available() > 0) {          //もしユーザーがデータ送信したら
      // read the bytes incoming from the client:
      char thisChar = client.read();          //ユーザーがネットワーク以上のデータを thisChar 変数へ送信
      // echo the bytes back to the client:
      server.write(thisChar);          // thisChar 変数を  server へ送信
      // echo the bytes to the server as well:
      Serial.write(thisChar);       // thisChar  変数出力
    }
  }
}
```

下図に示した通り、読者は本実験で Telnet 簡単チャットルームの、待機画面を見ることが可能である。

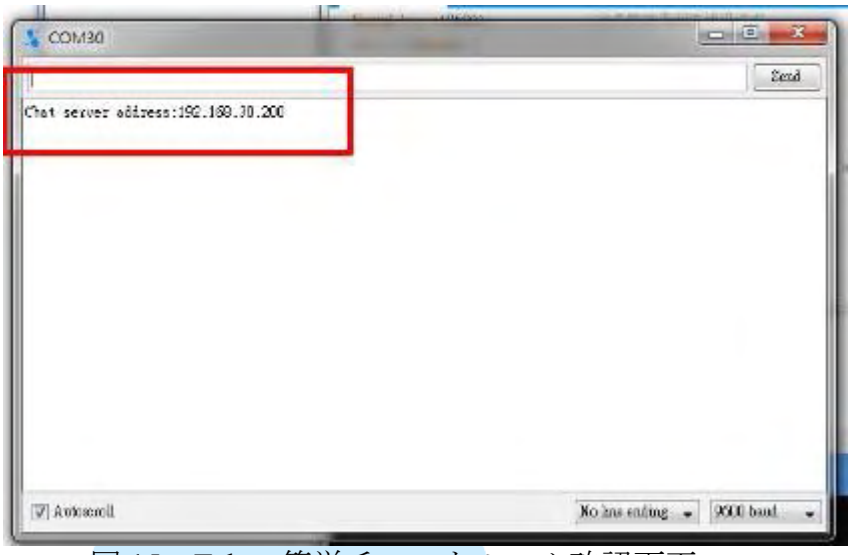


図 15　Telnet 簡単チャットルーム確認画面

下図に示した通り、読者は本実験 Telnet 簡単チャットルームを見ることが可能となるので、Putty 通信ソフトを使用し、あらかじめ設定した連線の画面を見てみよう。

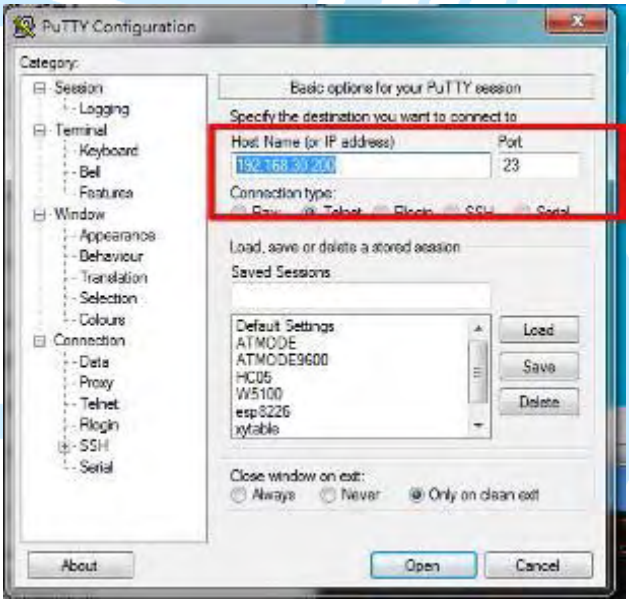


図 16 Telnet 簡単チャットルームあらかじめ設定した連線画面図

私たちは putty 通信ソフトを使用しチャットルームの機能を、下図に示した通り進行させ、簡単チャットルームにユーザー端画面をつないでみよう。

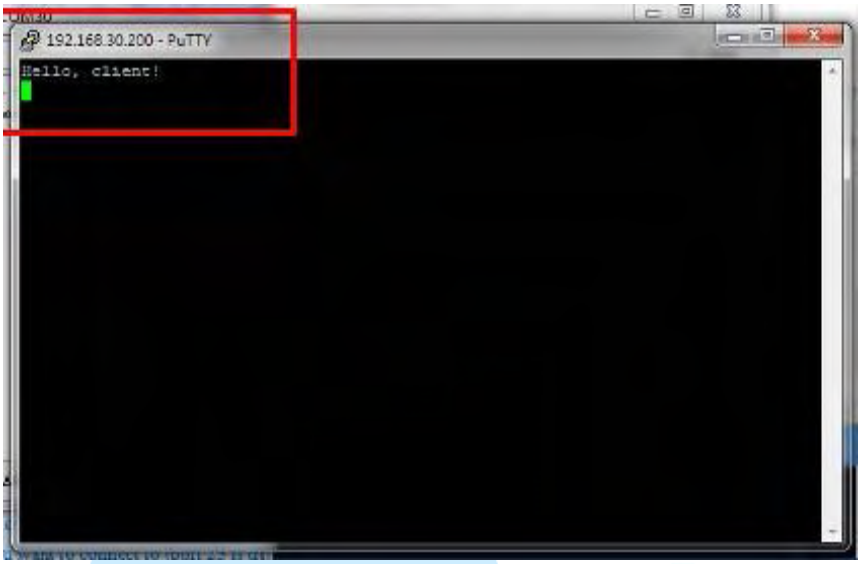


図 17 簡単チャットルームユーザー端画面連線図

下図に示した通り、Putty 通信ソフトを使用し、連線後、読者は本実験—Telnet 簡単チャットルームの Telnet 簡単チャットルームユーザー端連線の画面を見ることが可能となる。

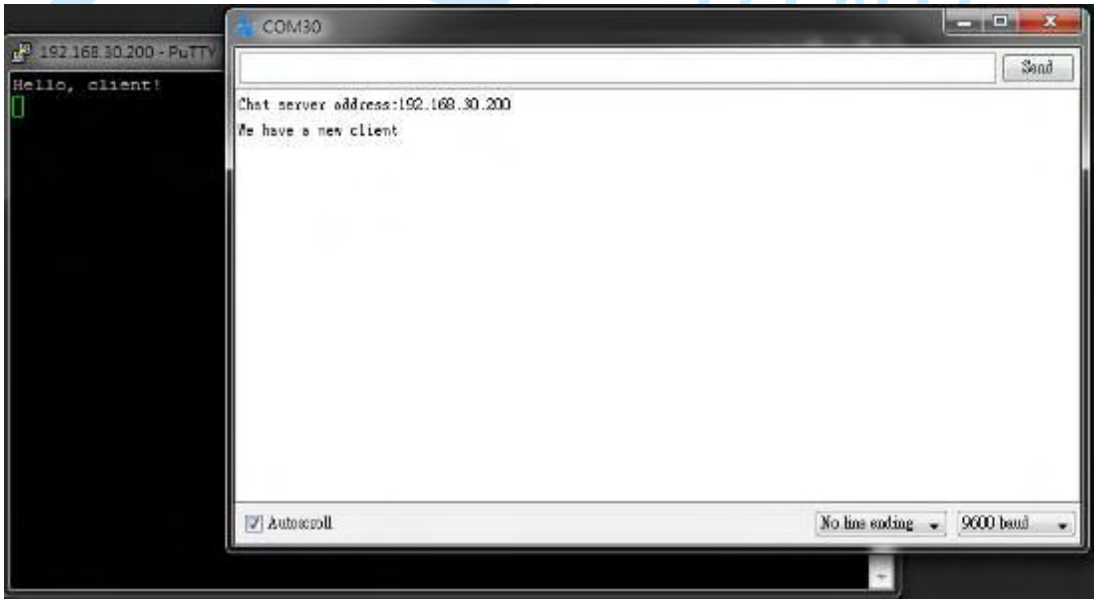


図 18 Telnet 簡単チャットルームユーザー端連線画面

8.Telnet 多人数版チャットルーム

先の節で、既に簡単かつ完全な『Telnet チャットルーム』を作り上げたが、このチャットルームはあまりにも小さく簡素で有る為、一人のユーザーしかチャットルームに入ることが出来ないので、この様な問題を改善する必要性が出てくる。

まず、下図に示したとおり、一般的な Cate 5 ネット回線を使用し、RJ45 を加え、86Duino EduCake を接続し、開発ボード側のネットインターフェイス(下図.(a))に接続し、ネット集線機のネットインターフェイス(下図.(b))にもう一方を接続し、86Duino EduCake 開発ボードの実態回路を完成させる。

(a). EduCake ネットワーク接続法　　(b). 集線器ネットワーク接続法

図 19 86Duino EduCake 開発ボードネットワーク接続法指示図

先に挙げた方法で、86Duino EduCake 開発ボードドライバープログラミング組込み後、86Duino EduCake 開発ボードの開発キットを開く：Sketch IDE が開発プログラミングを整合し、下図に示した Telnet チャットルームのテストプログラミングをコーディングし、86Duino EduCake 開発ボードを専門的な多人数チャットルームとして運用させよう。

表 7 Telnet 簡単多人数版チャットルームテストプログラム

(AdvancedChatServer2)

```
#include <SPI.h>
#include <Ethernet.h>          //   ネットワーク必須のものを使用

// 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
};          //この MAC ADRESS  をこの educake  の MAC ADDRESS と見なして使用
IPAddress ip(192, 168, 30, 200);          //あらかじめ設定したネットワーク IP アドレスを、読者は自由に変更使用可能
IPAddress dnServer(168, 95, 1, 1);       //あらかじめ設定した DNS サーバーを、本書では Hinet の DNS サーバーとし、読者は自由に変更使用可能
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);       //あらかじめ設定したゲートアドレス（つまり Rounter 或いは AP のアドレス）を、本書は作者がコーディングした環境のゲートアドレスとし、読者は自由に変更使用可能
// the subnet:
IPAddress subnet(255, 255, 255, 0);          //サブネット遮断、本書では Class C とする

// telnet defaults to port 23
EthernetServer server(23);          //サーバを宣告し、並びに Port23 を listening 通信ポートとして使用
boolean alreadyConnected ; // whether or not the client was connected previously  連線フラグ
boolean ConnectedFlag[10] ; // whether or not the client was connected previously 連線フラグ
int connectNo = 0 ;              //連線番号
   EthernetClient client ;       //ユーザーネットワーク 1 個生産
   EthernetClient Connectclient[10];          //10 個のユーザーネットワーク生産

void setup() {
```

```
  // initialize the ethernet device
  Ethernet.begin(mac, ip, gateway, subnet); //ネットワーク連線起動
  // start listening for clients
  server.begin();           // Port23 ネットワーク連線起動
  // Open serial communications and wait for port to open:
  Serial.begin(9600);           //画面通信速度宣告
    initConnectingFlag() ;   //   連線フラグ初期化

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

  Serial.print("Chat server address:");         // Chat server address:出力
  Serial.println(Ethernet.localIP());           //アドレス出力
  DisplayConnectingStatus() ;                        /連線状況表示
}

void loop() {
  // wait for a new client:
    client = server.available();           //ユーザーネットワーク生産

  // when the client sends the first byte, say hello:
  if (client) {        //ネットワークユーザ進入
    connectNo = 0 ;
    while (connectNo <10)        /10 組の連線ループ
    {
        if (!ConnectedFlag[connectNo]) { /もし新連線だとしたら
          // clead out the input buffer:
              Connectclient[connectNo] =client ;
            Connectclient[connectNo].flush();       //ユーザネットワークバッフ
ァーデータ更新
            Serial.println("We have a new client");          //新連線出力
            client.println("Hello, client!");         //  歓迎の言葉 Hello 送信
            ConnectedFlag[connectNo] = true;          //連線済み設定
            break ;
          }
        connectNo ++ ;
    }
```

```
  }
        connectNo = 0 ;
          while (connectNo <10)
    {
          if (!Connectclient[connectNo].connected())        //断線か否か診断
                  ConnectedFlag[connectNo] = false;       //連線フラグパラメータ
ー=false(未連線)

          if (Connectclient[connectNo].available() > 0)
          {    //
もしユーザーネットワークがデータ送信をしたら
            // clead out the input buffer:
            char thisChar = Connectclient[connectNo].read();          //ユーザネット
ワークが送信したデータ thisChar 変数読み取り
            // echo the bytes back to the client:
            Serial.print("Connect ");          //連線中出力
            Serial.print(connectNo);          //いくつかの連線出力
            Serial.print(":");        //出力":"
            server.write(thisChar);           // thisChar 変数を server へ送信
            // echo the bytes to the server as well:
            Serial.write(thisChar);       // thisChar 変数出力
          }
         connectNo ++ ;
    }

}

void initConnectingFlag()
{          //連線フラグパラメーター初期化
  for(connectNo=0 ; connectNo < 10 ; connectNo++)
  {
    ConnectedFlag[connectNo] = false ;          //連線フラグパラメーター=false
設定（未連線）
  }
}

void DisplayConnectingStatus()
{    //連線状況表示
```

```
    for(connectNo=0 ; connectNo < 10 ; connectNo++)
    {
      if (ConnectedFlag[connectNo])           //もし連線中なら
      {
            Serial.print("Connection ") ;        //当該連線中データ
Serial.print(connectNo) ;
            Serial.print(": Connected \n") ;
      }
      else       //もし連線中だとしたら
      {
            Serial.print("Connection ") ;        //当該表示される連線中データ
            Serial.print(connectNo) ;
            Serial.print(": Waiting Connecting \n") ;
      }

    // Connectclient[connectNo]    = server.available();        //ユーザーネットワーク
1 個表示
    }
}
```

下図に示した通り、読者は本実験 Telnet 多人数版チャットルームにより、ホストが開始した画面を見ることが可能となる。

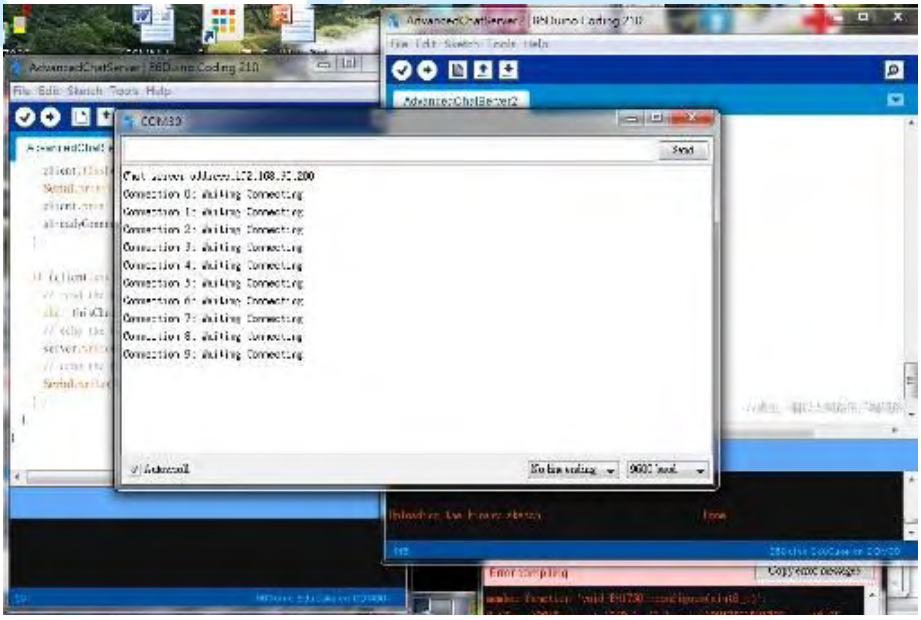


図 20 Telnet チャットルーム多人数版ホスト画面

下図に示した通り、Putty 通信ソフトを使用し、連線後、読者は本実験—Telnet 簡単チャットルームの Telnet 簡単チャットルームユーザー端連線の画面を見ることが可能となる。

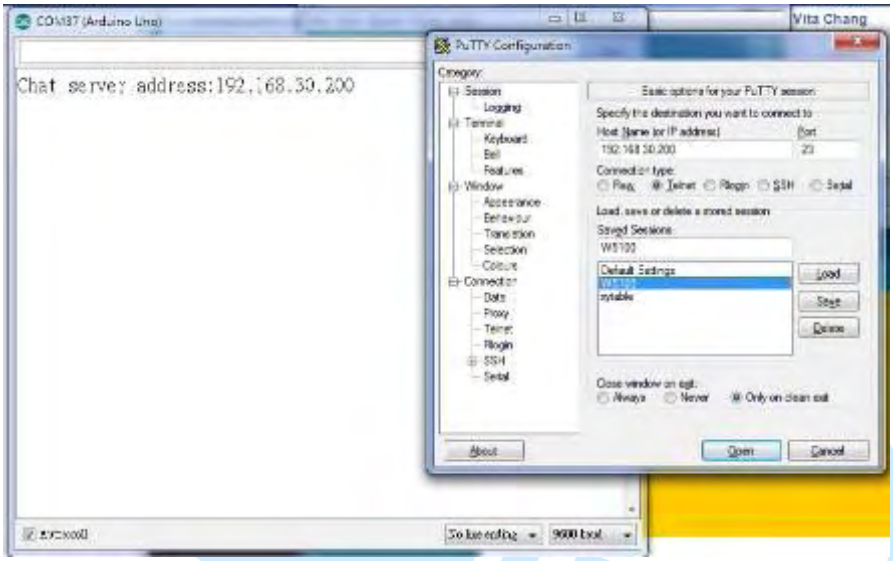


図 21 Telnet 簡単チャットルーム多人数版設定済み連線の画面

下図に示した通り、読者は本実験 Telnet 簡単チャットルームにより、Putty 通信ソフトを使用し 第一位の Telnet 簡単チャットルーム多人数版連線中の画面を見ることが可能である。

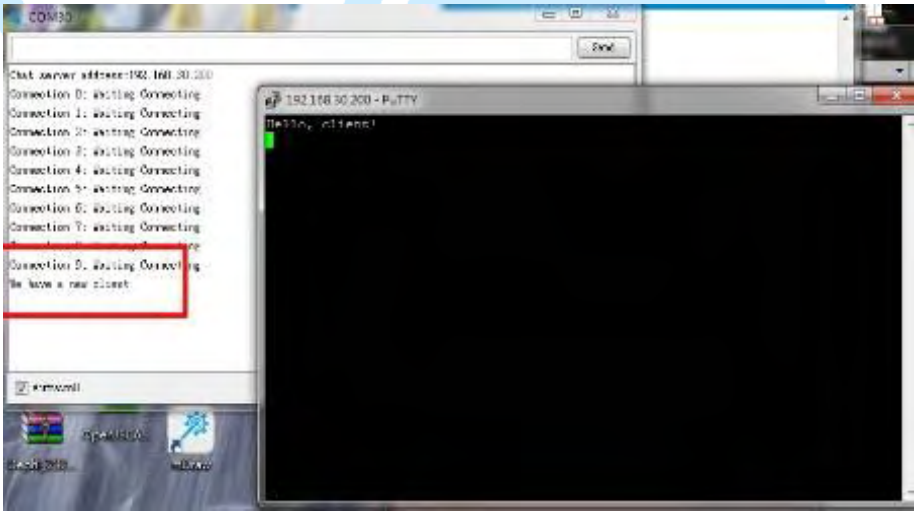


図 22 第一位 Telnet 簡単チャットルーム多人数版連線中画面

下図に示した通り、読者は本実験 Telnet 簡単チャットルームにより、Putty

通信ソフトを使用し 第二位の Telnet 簡単チャットルーム多人数版連線中の画面を見ることが可能である。

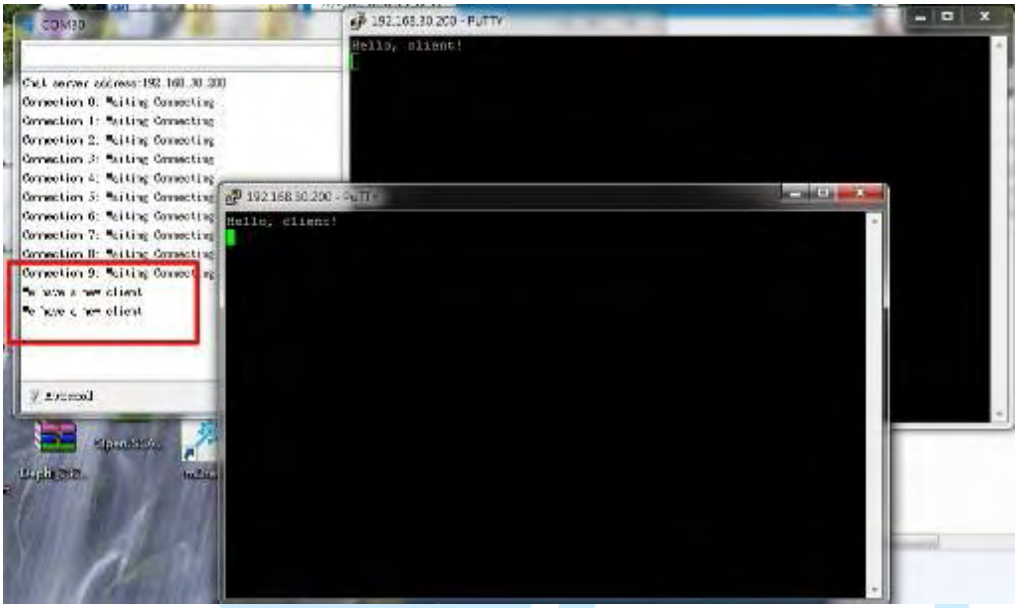


図 23 第二位 Telnet 簡単多人数版チャットルーム進行画面

9.温度コントロール

もし温度をテストしたいと思うなら、温度センサーを使用することで、その温度を測ることが出来るが、もし、湿度を測ろうとするなら、量測センサーを使用することで必要である。この様にして、様々なセンサーが必要であるが、本書では湿度患側モジュール(DHT11)を使用し、下図に示した通り、おもに DHT-11 を使用し湿度感測モジュールを作成する。

図 2 DHT11 湿度感測モジュール

本実験では DHT11 湿度感測モジュールを下図の様に採用し、DHT-11 湿度感測器は基本量測電気回路を搭載する必要が有る為、DHT11 湿度感測モジュールを使用し実験主体とし 並びに他の組み立て基本量測電気回路を説明することとする。

下図に示した通り、まず DHT11 湿度感測モジュールピンの接続法を参考にし、下の表を順守し、11 湿度感測モジュールピンを電気回路図に組み込んでいく。

図 25 DHT11 温湿度感測モジュールピン図

表 8 DHT11 温湿度感測モジュールピン

ピン	ピン説明	86Duino EduCake 開発モジュールピン
S	Vcc	電源 (+5V)
2	GND	EduCake GND
3	Signal	EduCake digital pin 7
		

データソース：Arduino プログラミング教学（常用モジュール編:Arduino Programming (37 Sensor Modules)(曹永忠, 許智誠, & 蔡英德, 2015b, 2015f)

先に挙げた方法で、86Duino EduCake 開発ボードドライバープログラミング組込み後、86Duino EduCake 開発ボードの開発キットを開く：Sketch IDE が開発プログラミングを整合し、下図に示した Telnet チャットルームのテストプログラミングをコーディングし、下図に示した通りの DHT11 温湿度感測モジュールテストプログラミングにより、DHT11 温湿度感測モジュールによる如何なる温度湿度も図りうることが可能である。

表 2 DHT11 温湿度感測モジュール(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);                    // 50us 待
```

```
機
                    delayMicroseconds(30);
// 高電位の持続時間判断、判定数字は ‘0’ もしくは ‘1’

                if(digitalRead(DHpin)==HIGH)
                    data |=(1<<(7-i));
//高位は前、低位は後ろ

                while(digitalRead(DHpin) == HIGH);          //  ‘1’ に
基づき、次の取得待機
                }
        }
        return data;
}

void start_test()
{
        digitalWrite(DHpin,LOW);                            //バスを引っ張り、開始
信号発信

        delay(30);                                                  // 遅延
時間は大よそ 18ms、検測機能シグナル開始；
        digitalWrite(DHpin,HIGH);
        delayMicroseconds(40);                              //観測機応答開始
；
        pinMode(DHpin,INPUT);
    while(digitalRead(DHpin) == HIGH);
          delayMicroseconds(80);                            //応答発信、低バス
80us；
        if(digitalRead(DHpin) == LOW);
              delayMicroseconds(80);                        //回線 80us デジタ
ル発送開始；

for(int i=0;i<4;i++)                                        //温湿度データ受
信、考慮せず；
        dat[i] = read_data();

          pinMode(DHpin,OUTPUT);
```

```
        digitalWrite(DHpin,HIGH);                                    //デジタル後の
回路発送完了、次のシグナル開始待機；
  }

void setup()
{
      Serial.begin(9600);
      pinMode(DHpin,OUTPUT);
}

void loop()
{
      start_test();
      Serial.print("Current humdity = ");
      Serial.print(dat[0], DEC);                                     //湿度の全パラ
メーター表示；
      Serial.print('.');
      Serial.print(dat[1],DEC);                                      //湿度の小デジ
タル表示；
      Serial.println('%');
      Serial.print("Current temperature = ");
      Serial.print(dat[2], DEC);                                     //温度の全デジタ
ル表示；
      Serial.print('.');
      Serial.print(dat[3],DEC);                                      //温度の小デジタ
ル表示；
      Serial.println('C');
      delay(700);
   }
```

参考データ：DMP 商店(http://shop.dmp.com.tw/INT/products/67)

下図に示したように、温度感測モジュールテストプログラミング結果画面を見ることが可能である。

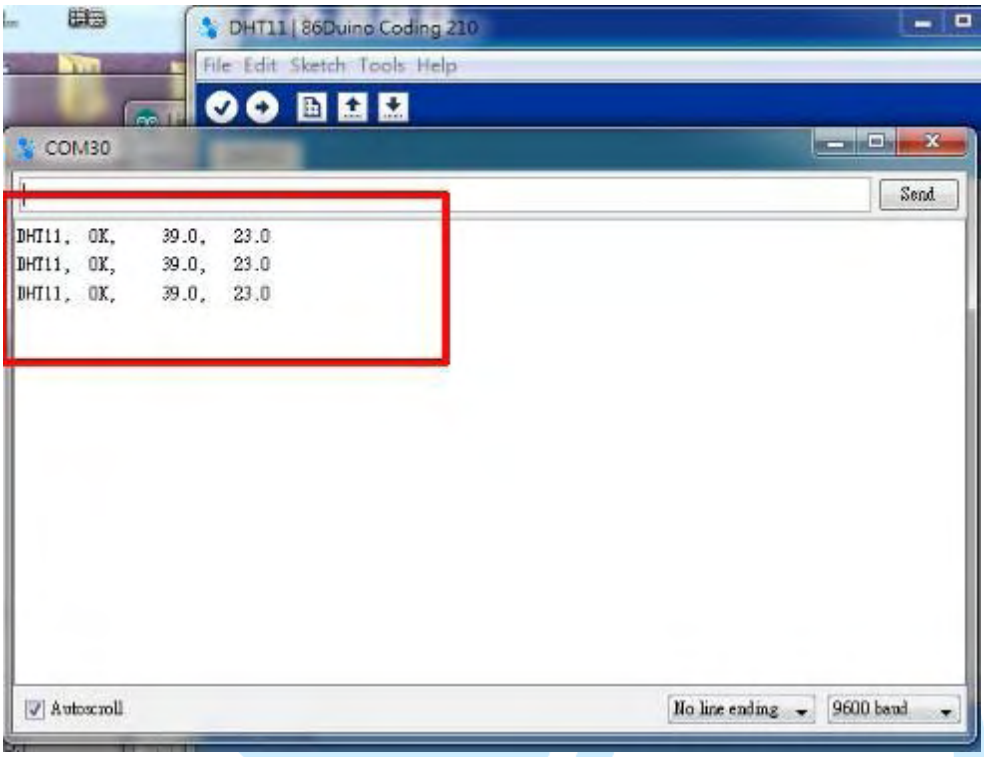


図 26 温度感測モジュールテストプログラミング結果画面

先のプログラミングは低級のコントロールシグナルを使用しているため、一般人では労力を使いすぎるので、作者はこれらプログラミングを Library の方式を使ってコーディングしようと思う。

10. 温湿度感測ウェブ画面

本章で先で述べた内容は、主に読者が 86Duino EduCake 開発ボードがネットワークソースをどのように使用するか、Telnet、Http の通信協定をどのように使用するか、ひいてはこれらネットワークソースを用いて簡単なネットワークサーバー、チャットサーバー等を作ることを中心としていたが、これらは本書の最後のプロジェクトとして基礎を打ち立てることとし、その前に、私たちはまず簡単な小プロジェクトを作ることとする。もし、温湿度読み取り装置を作ったとしたら、そのタイミングで既に作り上げた簡単かつ完全な『温湿度コントロールウェブサイト』を読み取り、そのようにして使用者はウェブブラウザを用いて温度データを取得することが可能となる。

まず、下図に示したとおり、一般的な Cate 5 ネット回線を使用し、RJ45 を加え、86Duino EduCake を接続し、開発ボード側のネットインターフェイス（下図.(a)）に接続し、ネット集線機のネットインターフェイス（下図.(b)）にもう一方を接続し、86Duino EduCake 開発ボードの実態回路を完成させる。

(a). EduCake ネットワーク接続法

(b). 集線器ネットワーク接続法

図 27 86Duino EduCake　開発ボードのネットワーク接続法指示図

先に挙げた方法で、86Duino EduCake 開発ボードドライバープログラミング組込み後、86Duino EduCake 開発ボードの開発キットを開く：Sketch IDE が開発プログラミングを整合し、下図に示した Telnet チャットルームのテストプログラ

ミングをコーディングし、下図に示した方法でネットウェブテストプログラミングをコントロールすることで、実作は完了し、作動することが可能となる。

表 10 温湿度制御ウェブテストプログラミング(TempMonitorServer)

```
#include <SPI.h>
#include <Ethernet.h>   //   ネットワーク必須のものを使用
#include <Wire.h>       //   ネット必須のものを使用
#include "dht.h"                 // DHT11 温湿度感測器必須のものを使用


#define DHT11_PIN 7         // DHT11 温湿度感測器通信ピン宣告
dht DHT;              //DHT11 温湿度感測器宣告
// 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
};        //この MAC ADRESS をこの educake  の MAC ADDRESS として使用
IPAddress ip(192, 168, 30, 200);          //あらかじめ設定したネットワーク IP アドレス，読者は自由に変更使用可能
IPAddress dnServer(168, 95, 1, 1);      //あらかじめ設定した DNS サーバー，本書では Hinet の DNS サーバーとし，読者は自由に変更使用可能
// the router's gateway address:
IPAddress gateway(192, 168, 30, 254);       //あらかじめ設定したゲートのアドレス(つまり Rounter 或いは AP のアドレス)，本書は作者がコーディングした環境のゲートアドレスとし、読者は自由に変更使用可能
// the subnet:
IPAddress subnet(255, 255, 255, 0);           //サブネット切断、本書は Class C とする

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);          //サーバ並びに Port 80 を通信ポートとして使用することを宣告
void setup() {
```

```
  // put your setup code here, to run once:
  Serial.begin(9600);                //コントロール画面の通信速度宣告
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  int chk = DHT.read11(DHT11_PIN);      //DHT11 温湿度感測器状態データ、並びに検査は使用可能か

    if ( chkDHT(chk) == 0)      //    chkDHT 関数使用、DHT11 の状態正常化否か，不正常 0，正常 1
        {
            Serial.println("ERROR on init DHT Sensor") ;          //
 DHT11 温湿度観測器不正常宣告
          while (true) ;          //プログラミング終了
        }

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip, dnServer, gateway, subnet);      //使 WEB サーバ起動、使用
  server.begin();          // Web サーバー起動
  Serial.print("server is at ");      //サーバーデータ出力
  Serial.println(Ethernet.localIP());              //サーバーIP アドレス出力


}

void loop() {

    EthernetClient client = server.available();          // WEB サーバーが Client 連線端起動
  if (client) {        // Client 連線端起動
    Serial.println("new client");          // "新連線"出力
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {          //もし連線成功なら
      if (client.available()) {          //連線の Client 連線端データ送信
        char c = client.read();          // Client 連先端のデータ変数  c 読み取り
```

```
            Serial.write(c);                    //変数 c 読み取り
            // 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) {          //変数 c 置き換え可能なら現
データを空とする
              // send a standard http response header
              client.println("HTTP/1.1 200 OK");          // Http  データ送信(必須)
              client.println("Content-Type: text/html");          // Http  データ送信(必須)

              client.println("Connection: close");    // the connection will be closed after
completion of the response     // Http  データ送信(必須)

              client.println("Refresh: 5");    // refresh the page automatically every 5 sec
      // Http  データ送信(必須)

              client.println();          // Http  データ送信(必須)

              client.println("<!DOCTYPE HTML>");          // Http  データ送信(必須)
              client.println("<html>");          // Http  データ送信(必須)
              // output the value of each analog input pin

                client.print("Humidity: ");            //目下の湿度提示字句送信
                client.print(DHT.humidity, 1);              //目下の湿度送信
                client.println("<br>");                  //ネット交換機キー送信
                client.print("Temperature: ");          //目下の温度提示字句送信
                client.print(DHT.temperature, 1);          //目下の温度送信
                client.println("<br>");                  //ネット交換機キー送信

              client.println("</html>");          //ネット Tag データ送信(必須)
              break;
            }
            if (c == '\n') {          //新たな一行
              // you're starting a new line
              currentLineIsBlank = true;
            }
            else if (c != '\r') {          //もし交換キーでなくば
              // you've gotten a character on the current line
```

```
                currentLineIsBlank = false;
            }
          }
        }
        // give the web browser time to receive the data
        delay(1);       //一秒遅延
        // close the connection:
        client.stop();          //連線の Client 連先端開閉(必須)
        Serial.println("client disconnected");                  //連線断線送信
    }


    delay(2000);            //遅延 2000 秒
}

unsigned int chkDHT( int chk )          //DHT11 温湿度感測器状態データ正確か表
示
{
     switch (chk)           // chk 変数チェック;多重検査
    {
        case DHTLIB_OK:             //もし ok 状態なら
                    Serial.println("DHT init is OK,\t");            // ok 状態出力
                    return 1;
        case DHTLIB_ERROR_CHECKSUM:       //もし数値が不正確であれば
                    Serial.println("DHT Checksum error,\t");          //数値不正確状
態出力
                    return 0;
        case DHTLIB_ERROR_TIMEOUT:          //もし固定時間が正確な時間を読
み取れなければ
                    Serial.println("DHT Time out error,\t");      //固定時間が正確な
データ読み取れないことを出力
                  return 0;
        default:                //ミス
                    Serial.println("DHT Unknown error,\t");           //ミス
                  return 0;
    }

}
```

下図に示した通り、読者は本実験により、温湿度コントロールネットワーク画面を見ることが可能となる。

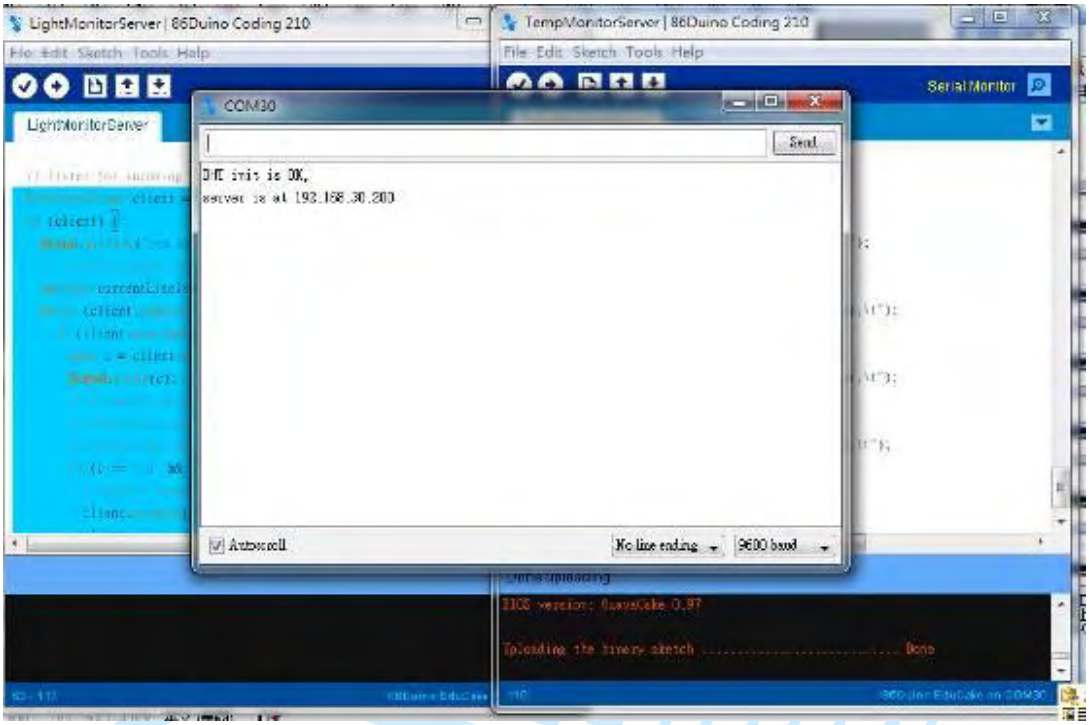


図 28 温湿度器主画面

下図に示したように、私たちはこれらを使用し、温湿度に関する様々なデータを取得することが可能となる。

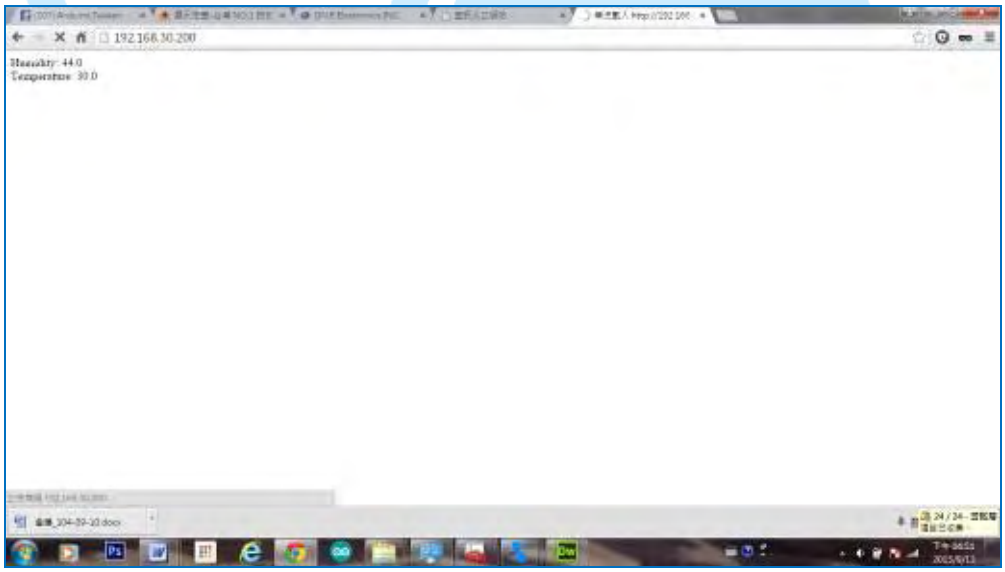


図 29 温湿度器主画面

下図に示したように、温湿度感測器がウェブブラウザに応答することし、その HTML により、読者は温湿度装置による温湿度データを使用することが可能となる。

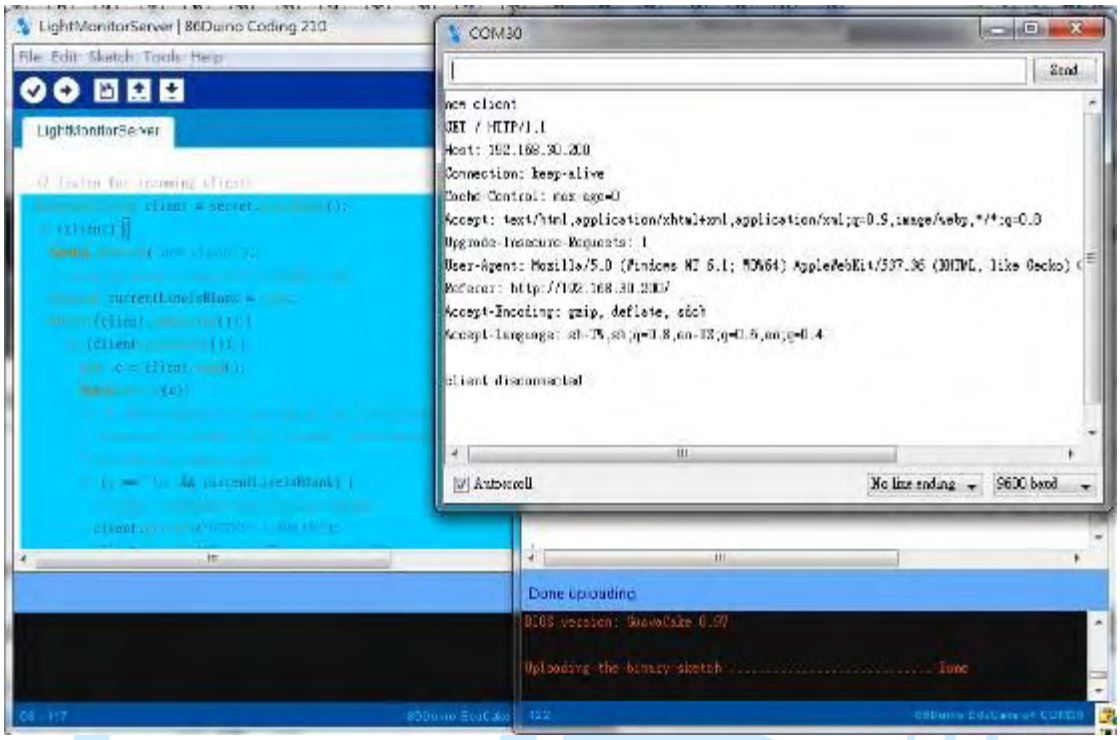


図 30 温湿度感測器ウェブブラウザ表示

11. 末尾として

本書では主に 86Duino EduCake 開発ボードが内蔵するイーサネットを使用することで可能となる、温湿度データの取得、転送、並びにウェブブラウザに表示されるデータについて論じた。