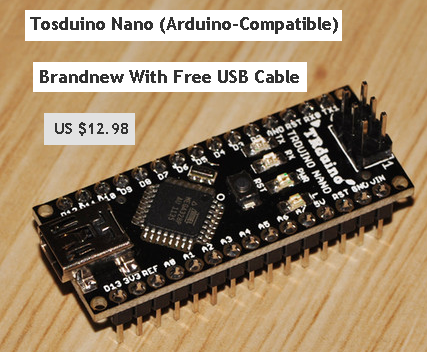
**ARDUINO WEBSERVER from low cost boards**

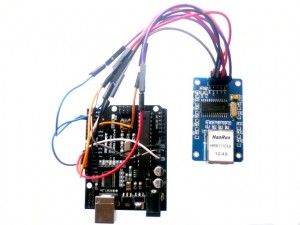


A very reasonable option is to use Arduino nano. That already includes USB serial connection for programming.

Cheap ENC28J60 boards exist. (Cheaper then ordering individual parts).  
Unfortunately no 3.3 V regulator onboard. Can use 2 diodes in series to make ca 3.3 V from 5V. Or have 3.3 V regulator. Taking 3.3V from FTDI chip makes whole thing unreliable in long term use.  
Some example of usage:   
<http://voltsandbytes.com/simple-sensor-webserver/>

No need to solder. Can connect to Ethernet board using 9 wires with connectors for breadboard.

Webservers made using Arduino are very reliable and do not hang like some more advanced Linux boards.

Unfortunately presently there are no cheap priced boards with both Atmega 328 and ENC28J60 chips,   
but may be in a year or so will appear on Ebay.

There are compatibility issues comparing to original Arduino Ethernet shields. Even worse, 3 different Ethernet driver version exist which are not compatible with each other. Luckily there is examples folder with each driver version so that one can see what can get.

<http://blog.thiseldo.co.uk/?p=504>

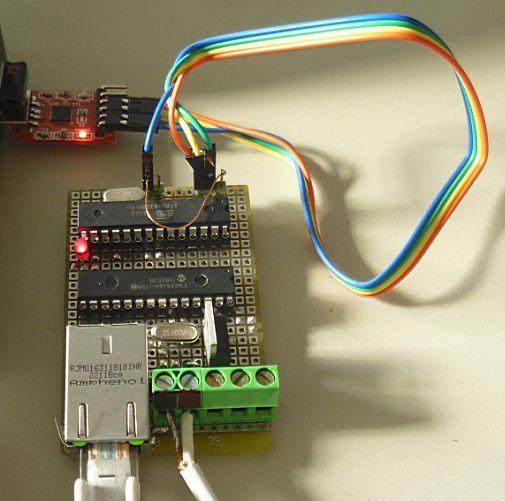
DNS recogntion worked, but DHCP client did not.

For example could program sensor value upload to Pachube, but example for remote control via Pachube cpould not get working.

**Links**

<http://www.komputer.de/wordpress/archives/497>  
<http://wiki.london.hackspace.org.uk/view/Project:Nanode>  
<http://wiki.london.hackspace.org.uk/view/Project:Ethernet_Arduino>

I switched from AVR C programming to Arduino, because I already had written programs for ENC28J60 Ethernet shield that could be used to drive relays for “Internet controlled power socket”.  
In the program below I implemented saving to EEPROM status of the two pins that can be controlled via Internet.



To switch from TuxGraphics to Arduino schematics was easy. Had to add a 16 MHz quartz oscillator with 18 pF capacitors. Actually Arduino should not run from 3.3V at 16 MHz but it still does. All other lines stayed same SCK, SI SO INT CS. Put in a sockedt ATMEGA328 with Arduino bootloaded chip.

CP2102 adapter. RX, TX, GND. Arduino bootloaded chip powered from 3.3 V .

In menu choose board Arduino Pro mini 16 MHz 328.

When Arduino SDK prints line “uploading…” , should short to GND reset pin for short while.

Code for Arduino webserver to switch power outlets :

#include "EEPROM.h"

#include "etherShield.h"

// Two IP controlled power outlets via Arduino and ENJ2860 shield.

// please modify the following two lines. mac and ip have to be unique

// in your local area network. You can not have the same numbers in

// two devices:

static uint8\_t mymac[6] = {0x54,0x55,0x58,0x10,0x00,0x24};

static uint8\_t myip[4] = {192,168,0,14};

static char baseurl[]="http://192.168.0.14/";

static uint16\_t mywwwport =80; // listen port for tcp/www (max range 1-254)

#define BUFFER\_SIZE 1300

static uint8\_t buf[BUFFER\_SIZE+1];

#define STR\_BUFFER\_SIZE 22

static char strbuf[STR\_BUFFER\_SIZE+1];

EtherShield es=EtherShield();

// prepare the webpage by writing the data to the tcp send buffer

uint16\_t print\_webpage(uint8\_t \*buf, byte on\_off=0);

int8\_t analyse\_cmd(char \*str);

// OUTPUTS to relay driving transnistors. First check with LEDs.

#define LED\_PIN1 4

#define LED\_PIN2 5

void setup(){

/\*initialize enc28j60\*/

es.ES\_enc28j60Init(mymac);

es.ES\_enc28j60clkout(2); // change clkout from 6.25MHz to 12.5MHz

delay(10);

/\* Magjack leds configuration, see enc28j60 datasheet, page 11 \*/

// LEDA=greed LEDB=yellow

//

// 0x880 is PHLCON LEDB=on, LEDA=on

// enc28j60PhyWrite(PHLCON,0b0000 1000 1000 00 00);

es.ES\_enc28j60PhyWrite(PHLCON,0x880);

delay(500);

//

// 0x990 is PHLCON LEDB=off, LEDA=off

// enc28j60PhyWrite(PHLCON,0b0000 1001 1001 00 00);

es.ES\_enc28j60PhyWrite(PHLCON,0x990);

delay(500);

//

// 0x880 is PHLCON LEDB=on, LEDA=on

// enc28j60PhyWrite(PHLCON,0b0000 1000 1000 00 00);

es.ES\_enc28j60PhyWrite(PHLCON,0x880);

delay(500);

//

// 0x990 is PHLCON LEDB=off, LEDA=off

// enc28j60PhyWrite(PHLCON,0b0000 1001 1001 00 00);

es.ES\_enc28j60PhyWrite(PHLCON,0x990);

delay(500);

//

// 0x476 is PHLCON LEDA=links status, LEDB=receive/transmit

// enc28j60PhyWrite(PHLCON,0b0000 0100 0111 01 10);

es.ES\_enc28j60PhyWrite(PHLCON,0x476);

delay(100);

//init the ethernet/ip layer:

es.ES\_init\_ip\_arp\_udp\_tcp(mymac,myip,80);

pinMode(LED\_PIN1, OUTPUT);

digitalWrite(LED\_PIN1, LOW); // switch on LED

if (EEPROM.read(1)==1){digitalWrite(LED\_PIN1, HIGH);}

pinMode(LED\_PIN2, OUTPUT);

digitalWrite(LED\_PIN2, LOW); // switch on LED

if (EEPROM.read(2)==1){digitalWrite(LED\_PIN2, HIGH);}

}

void loop(){

uint16\_t plen, dat\_p;

int8\_t cmd;

byte on\_off = 0;

plen = es.ES\_enc28j60PacketReceive(BUFFER\_SIZE, buf);

/\*plen will ne unequal to zero if there is a valid packet (without crc error) \*/

if(plen!=0){

// arp is broadcast if unknown but a host may also verify the mac address by sending it to a unicast address.

if(es.ES\_eth\_type\_is\_arp\_and\_my\_ip(buf,plen)){

es.ES\_make\_arp\_answer\_from\_request(buf);

return;

}

// check if ip packets are for us:

if(es.ES\_eth\_type\_is\_ip\_and\_my\_ip(buf,plen)==0){

return;

}

if(buf[IP\_PROTO\_P]==IP\_PROTO\_ICMP\_V && buf[ICMP\_TYPE\_P]==ICMP\_TYPE\_ECHOREQUEST\_V){

es.ES\_make\_echo\_reply\_from\_request(buf,plen);

return;

}

// tcp port www start, compare only the lower byte

if (buf[IP\_PROTO\_P]==IP\_PROTO\_TCP\_V&&buf[TCP\_DST\_PORT\_H\_P]==0&&buf[TCP\_DST\_PORT\_L\_P]==mywwwport){

if (buf[TCP\_FLAGS\_P] & TCP\_FLAGS\_SYN\_V){

es.ES\_make\_tcp\_synack\_from\_syn(buf); // make\_tcp\_synack\_from\_syn does already send the syn,ack

return;

}

if (buf[TCP\_FLAGS\_P] & TCP\_FLAGS\_ACK\_V){

es.ES\_init\_len\_info(buf); // init some data structures

dat\_p=es.ES\_get\_tcp\_data\_pointer();

if (dat\_p==0){ // we can possibly have no data, just ack:

if (buf[TCP\_FLAGS\_P] & TCP\_FLAGS\_FIN\_V){

es.ES\_make\_tcp\_ack\_from\_any(buf);

}

return;

}

if (strncmp("GET ",(char \*)&(buf[dat\_p]),4)!=0){

// head, post and other methods for possible status codes see:

// http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

plen=es.ES\_fill\_tcp\_data\_p(buf,0,PSTR("HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n<h1>200 OK</h1>"));

goto SENDTCP;

}

if (strncmp("/ ",(char \*)&(buf[dat\_p+4]),2)==0){

plen=print\_webpage(buf, on\_off);

goto SENDTCP;

}

cmd=analyse\_cmd((char \*)&(buf[dat\_p+5]));

on\_off=-1;

if (cmd==0){

on\_off=0;

digitalWrite(LED\_PIN1, LOW); // switch on LED

EEPROM.write(1, 0);

digitalWrite(LED\_PIN2, LOW); // switch on LED

EEPROM.write(2, 0);

}

else if (cmd==1){

on\_off=1;

digitalWrite(LED\_PIN1, HIGH); // switch off LED

EEPROM.write(1, 1);

digitalWrite(LED\_PIN2, LOW); // switch on LED

EEPROM.write(2, 0);

}

else if (cmd==2){

on\_off=2;

digitalWrite(LED\_PIN1, LOW); // switch off LED

EEPROM.write(1, 0);

digitalWrite(LED\_PIN2, HIGH); // switch on LED

EEPROM.write(2, 1);

}

else if (cmd==3){

on\_off=3;

digitalWrite(LED\_PIN1, HIGH); // switch off LED

EEPROM.write(1, 1);

digitalWrite(LED\_PIN2, HIGH); // switch off LED

EEPROM.write(2, 1);

}

plen=print\_webpage(buf, on\_off);

plen=print\_webpage(buf, on\_off);

SENDTCP: es.ES\_make\_tcp\_ack\_from\_any(buf); // send ack for http get

es.ES\_make\_tcp\_ack\_with\_data(buf,plen); // send data

}

}

}

}

// The returned value is stored in the global var strbuf

uint8\_t find\_key\_val(char \*str,char \*key)

{

uint8\_t found=0;

uint8\_t i=0;

char \*kp;

kp=key;

while(\*str && \*str!=' ' && found==0){

if (\*str == \*kp){

kp++;

if (\*kp == '\0'){

str++;

kp=key;

if (\*str == '='){

found=1;

}

}

}else{

kp=key;

}

str++;

}

if (found==1){

// copy the value to a buffer and terminate it with '\0'

while(\*str && \*str!=' ' && \*str!='&' && i<STR\_BUFFER\_SIZE){

strbuf[i]=\*str;

i++;

str++;

}

strbuf[i]='\0';

}

return(found);

}

int8\_t analyse\_cmd(char \*str)

{

int8\_t r=-1;

if (find\_key\_val(str,"cmd")){

if (\*strbuf < 0x3a && \*strbuf > 0x2f){

// is a ASCII number, return it

r=(\*strbuf-0x30);

}

}

return r;

}

uint16\_t print\_webpage(uint8\_t \*buf, byte on\_off)

{

uint16\_t plen;

plen=es.ES\_fill\_tcp\_data\_p(buf,0,PSTR("HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n"));

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<center><p><h3>IP controlled 220V power outlets <br>using Arduino with ENC28J60 ethernet shield</h3></p> "));

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<hr><br><form METHOD=get action=\""));

plen=es.ES\_fill\_tcp\_data(buf,plen,baseurl);

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("\">"));

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<h1><font color=\"#00FF00\"> "));

if(digitalRead(4)==0)

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("LAMP = OFF"));

else

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("LAMP = ON"));

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<BR><BR>"));

if(digitalRead(5)==0)

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("PC = ON"));

else

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("PC = OFF"));

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR(" </font></h1><br> ") );

if(digitalRead(4)==0){

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<br><input type=hidden name=cmd value=1>"));

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<input type=submit value=\"Switch LAMP ON\"></form><hl>"));

}

else {

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<br><input type=hidden name=cmd value=0>"));

plen=es.ES\_fill\_tcp\_data\_p(buf,plen,PSTR("<input type=submit value=\"Switch LAMP OFF\"></form><hl>"));

}

return(plen);

}