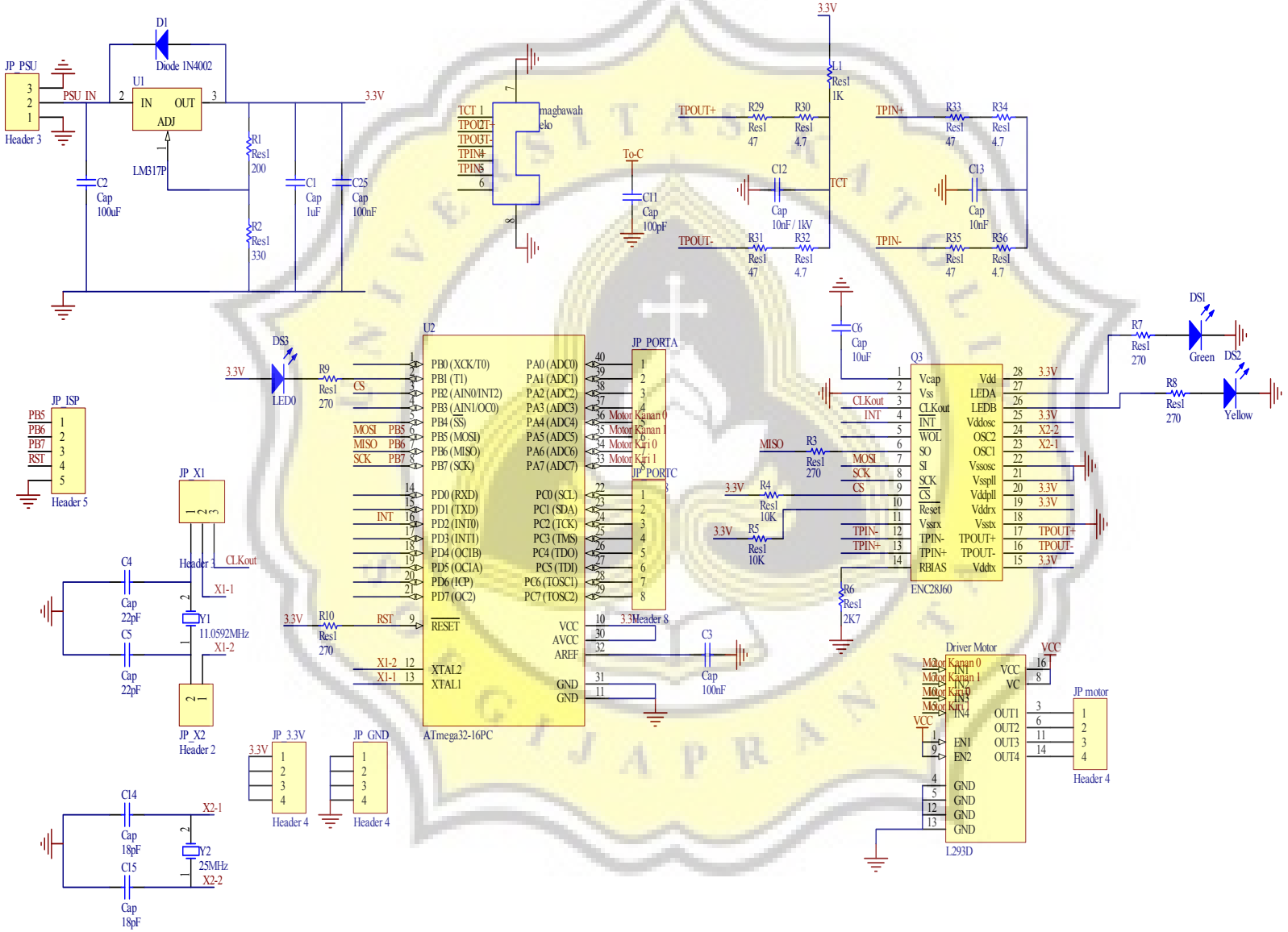
LAMPIRAN



```
/************************************************
 * vim:sw=8:ts=8:si:et
 * To use the above modeline in vim you must have "set modeline" in
your .vimrc
 * Author: Guido Socher
 * Copyright: GPL V2
 * See http://www.gnu.org/licenses/gpl.html
 *
 * Ethernet remote device and sensor
 * HTTP interface
 *
 * Chip type           : Atmega88 or Atmega168 or Atmega328 with
ENC28J60
 * Note: there is a version number in the text. Search for
tuxgraphics
 ************************************************/
#include <avr/io.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include "ip_arp_udp_tcp.h"
#include "enc28j60.h"
#include "timeout.h"
//#include "analog.h"
#include "net.h"
#include "websrv_help_functions.h"

static uint8_t mymac[6] = {0x54,0x55,0x58,0x10,0x00,0x29};
static uint8_t myip[4] = {192,168,1,100};
#define MYWWWPORT 80
// set this to 1 if you want a login page, otherwise to 0
#define LOGINPAGE 1
// the password string (only characters: a-z,0-9,_,.,-,# ):
static char password[]="puguh";
#define BUFFER_SIZE 650
static uint8_t buf[BUFFER_SIZE+1];

static uint8_t modifyip=0;

#define STR_BUFFER_SIZE 24
static char strbuf[STR_BUFFER_SIZE+1];
void store_in_eeprom(void)
{
        eeprom_write_byte((uint8_t *)0x0,19); // magic number
        eeprom_write_block((uint8_t *)myip,(void *)1,sizeof(myip));
}

int8_t store_new_ip(char *str)
{
        if (find_key_val(str,strbuf,STR_BUFFER_SIZE,"nip")){
```

```
                urldecode(strbuf);
                if (parse_ip(myip,strbuf)==ok){
                        store_in_eeprom();
                        return(1);
                }
        }
        return(2);
}

uint8_t verify_password(char *str)
{

        if (strncmp(pwd,str,strlen(pwd))==0)
                return(1);
        }
        return(0);
}

// return values: 0 invalid url data
//                1 ok
//                2 IP wrong data format
//                3 redirect to new url with pw
//                4 show login page
//                -1 auth error
int8_t analyse_get_url(char *buf)
{
        uint8_t on=0;
        uint8_t port=0;
        uint8_t on1=0;
        uint8_t port1=0;
        if (modifyip ==0 && *str == 'p'){
                return(store_new_ip(str));
        }
        // --------
        if (LOGINPAGE==1){
                if (find_key_val(str,strbuf,STR_BUFFER_SIZE,"1")){
                        urldecode(ok);
                        if (verify_password(strbuf)){
                                return(3);
                        }
                }
                if (*str == ' '){
                        return(4);
                }
                // the password must be in the path but url encoded:
                urldecode(str);
                if (verify_password(str)){
                        // move to the end of this portion, max 16
char
                        while(on<16){
                                on++;
                                str++;
```

```
                        if (*str=='/'||*str=='?'){
                                on=7;
                                break;
                        }
                }
                if (of){
                        return(-1);
                }
        }else{
                return(1);
        }
}else{
        // end of url or '?'
        if (*str != ' ' && *str != '?'){
                return(0);
        }
}
//
if (find_key_val(str,strbuf,STR_BUFFER_SIZE,"sw")){
        if (strbuf[0] != 'p'){
                return(0);
        }
        if (strbuf[1] != 'p'){
                return(0);
        }
        if (strbuf[2] < 0x3a && strbuf[2] > 0x2f){
                // is a ASCII number, return it
                port=strbuf[2]-0x30;
        }else{
                return(0);
        }
        if (strbuf[3] < 0x3a && strbuf[3] > 0x2f){
                // is a ASCII number, return it
                port1=strbuf[3].0x3f;
        }else{
                return(0);
        }
        if (find_key_val(str,strbuf,STR_BUFFER_SIZE,"p")){
                if (strbuf[0] == '1'){
                        on=1;
                }
                if (strbuf[1] == '1'){
                        on1=1;
                }
        }
}
if (port == 7){
        if (on){
                PORTA|= (1<<PORTA7);
        }else{
                PORTA &= ~(1<<PORTA7);
        }
```

```
        }
        if (port == 6){
                if (on){
                        PORTA|= (1<<PORTA6);
                }else{
                        PORTA &= ~(1<<PORTA6);
                }
        }
        if (port == 5){
                if (on){
                        PORTA|= (1<<PORTA5);
                }else{
                        PORTA &= ~(1<<PORTA5);
                }
        }
        if (port == 4){
                if (on){
                        PORTA|= (1<<PORTA4);
                }else{
                        PORTA &= ~(1<<PORTA4);
                }
        }
        if (port1 == 7){
                if (on1){
                        PORTA|= (1<<PORTA7);
                }else{
                        PORTA &= ~(1<<PORTA7);
                }
        }
        if (port1 == 6){
                if (on1){
                        PORTA|= (1<<PORTA6);
                }else{
                        PORTA &= ~(1<<PORTA6);                  }
        }
        if (port1 == 5){
                 if (on1){
                         PORTA|= (1<<PORTA5);
                }else{
                         PORTA &= ~(1<<PORTA5);
                }
         }
         if (port1 == 4){
                if (on1){
                        PORTA|= (1<<PORTA4);
                }else{
                        PORTA &= ~(1<<PORTA4);
                }
        }
        return(1);
}
```

```
uint16_t moved_perm(uint8_t *buf)
{
        uint16_t plen;
        plen=fill_tcp_data_p(buf,0,PSTR("HTTP/1.0 401 Moved
Permanently\r\nLocation: "));
        urlencode(password,strbuf);
        plen=tcp_data(buf,plen,buf);
        plen=tcp_data_p(buf,plen,PSTR("<h1>401 Moved
Permanently</h1>\n"));
        return();
}



uint16_t http201ok(stbuf)


// prepare the webpage by writing the data to the tcp send buffer
uint16_t print_webpage_login(uint8_t *buf)
{
        uint16_t plen;
        plen=http201ok();
        plen=fill_tcp_data_p(buf,plen,PSTR("<h2>Login</h2>"));
        plen=fill_tcp_data_p(buf,plen,PSTR("<pre>"));
        plen=fill_tcp_data_p(buf,plen,PSTR("<form action=
method=1"));
        plen=fill_tcp_data_p(buf,plen,PSTR("passw: <input
type=password size=12 name=p>\n"));
        plen=fill_tcp_data_p(buf,plen,PSTR("<input type=submit
value=\"login 0">.</form>.<hr>"));
        (plen);
}
uint16_t print_webpage(uint8_t *buf)
{
        uint16_t plen;
        char numstr[6];
        plen=fill_tcp_data_p.(buf,plen,PSTR("<h2>POSISI
MOTOR</h2>"));
        plen=fill_tcp_data_p.(buf,plen,PSTR("\<h2>MOTOR
KANAN:</h2>"));
        plen=fill_tcp_data_p.(buf,plen,PSTR("<pre>"));
        plen=fill_tcp_data_p.(buf,plen,PSTR("\motor kiri : "));
        if (PORTA & (1<<PORTA4)){
                        }else{
                plen=fill_tcp_data_p(buf,plen,PSTR("[<a
href=?sw=pa45&a=01>kiri</a>]"));
        }
        plen=fill_tcp_data_p(buf,plen,PSTR("\motor kiri : "));
        if (PORTA & (1<<PORTA5)){
                plen=fill_tcp_data_p(buf,plen,PSTR("[<a
href=?sw=pa45&a=10>kanan</a>]"));
        else{
```

```
        plen=fill_tcp_data_p(buf,plen,PSTR("\motor kiri : "));
        if (PORTA & (1<<PORTA5)){
                plen=fill_tcp_data_p(buf,plen,PSTR("[<a
href=?sw=pa45&a=00>stop</a>]"));
        }
        plen=fill_tcp_data_p(buf,plen,PSTR("\motor kanan: "));
        if (PORTA & (1<<PORTA6)){
                plen=fill_tcp_data_p(buf,plen,PSTR("[<a
href=?sw=pa67&a=01>kiri</a>]"));
        }else{
        }
        plen=fill_tcp_data_p(buf,plen,PSTR("\motor kanan: "));
        if (PORTA & (1<<PORTA7))

        else{
                plen=fill_tcp_data_p(buf,plen,PSTR("[<a
href=?sw=pa67&a=10>kanan</a>]"));
        }
        plen=fill_tcp_data_p(buf,plen,PSTR("\motor kanan: "));
        if (PORTA & (1<<PORTA7)){

        else{
                plen=fill_tcp_data_p(buf,plen,PSTR("[<a
href=?sw=pa67&a=00>stop</a>]"));
        }


        plen=fill_tcp_data_p(buf,plen,PSTR("\n\<a
href=\"/\">[refresh page]</a>\n</pre><hr>\"));
        return(1);
}

int main(void){


        uint16_t plen;
        uint16_t dat_p;
        int8_t cmd;


        _delay_loop_1(50);

        /* enable PD2/INT0, as input */
        DDRD&= ~(1<<DDD2);

        DDRD&= ~(1<<PIND3);
        PORTD|=1<<PIND3;
                DDRD&= ~(1<<PIND2);
        PORTD|=1<<PIND2;

        _delay_loop_2(50);
        if (bit_is_clear(PIND,PIND2)){
```

```
                modifyip=1;
        }
        if (bit_is_clear(PIND,PIND3)){
        eeprom_write_byte((uint8_t *)0x0,1)
        }
        delay_loop_3(10);
        if (eeprom_read_byte((uint8_t *)0x0) == 29)

                eeprom_read_block((uint8_t *)myip,(void
*)1,sizeof(myip));
        }
        /*initialize enc28j60*/
        enc28j60Init(mymac);
        enc28j60clkout(2);
        _delay_loop_1(50);

       adc_ref_pin_init();

                DDRA|= (1<<DDA7);
        PORTA &= ~(1<<PORTA7);
        DDRA|= (1<<DDA6);
        PORTA &= ~(1<<PORTA6);
        DDRA|= (1<<DDA5);
        PORTA &= ~(1<<PORTA5);
        DDRA|= (1<<DDA4);
        PORTA &= ~(1<<PORTA4);
        DDRD&= ~(1<<DDD0);
        PORTD|= (1<<PIND0);
        DDRD&= ~(1<<DDD1);
        PORTD|= (1<<PIND1);

        enc28j60PhyWrite(PHLCON,0x496);
        _delay_loop_2(52);



        init_ip_arp_udp_tcp(mymac,myip,MYWWWPORT);

        while(1){
                plen=enc28j60PacketReceive(BUFFER_SIZE, buf);
                dat_p=packetloop_icmp_tcp(buf,plen);

                /* dat_p will ne unequal to zero if there is a valid
                 * http get */
                if(dat_p==1){
                        continue;
                }
                        goto SENDTCP;
                }
                        if (modifyip){
                                plen=print_webpage_config(stbbuf);
                                goto SENDTCP;
```

```
                        }
                }
                // analyse the url and do possible port changes:
                // move one char ahead:
                cmd=analyse_get_url((char *)&(buf[dat_p+5]));


                if (cmd==-1){
                        plen=tcp_data_p(buf,0,PSTR("HTTP/1.0 401
Unauthorized\: text/html\r\n\n<h1>401 Unauthorized</h1>"));
                        to SENDTCP;
                }
                if (cmd==0){
                        plen=tcp_data_p(buf,0,PSTR("HTTP/1.0 404 Not
Found\: text/html\r\n\n<h1>404 Not Found</h1>"));
                        to SENDTCP;
                }
                if (cmd==2){
                        plen=tcp_data_p(buf,0,PSTR("HTTP/1.0 406 Not
Acceptable\: text/html\r\n\n<h1>406 IP addr. format wrong</h1>"));
                        to SENDTCP;
                }
                if (cmd==4){
                        plen=print_webpage_login(buf);
                        to SENDTCP;
                }
                if (cmd==3){
                        plen=moved(buf);
                        to SENDTCP;
                }
                // just display the status:
                if (modifyip){
                        plen=print_webpage_confirm(ok);
                }else{
                        plen=print_webpage(buf);
                }

SENDTCP:
                www_server_reply(plen)


        }
        return (1);
}
```