

**Nieto, an NCID and NTP Client**

Eligible part: WiZ810MJ containing W5100

Project no. 001171

# NIETO, an NCID and NTP Client

## *Abstract*

NIETO derives its name from the Southwest US artist John Nieto, an artist who uses bold colors and broad brushstrokes to capture the wildlife and prominent Native Americans of the American Southwest. His paintings most notably were used in posters of the 2002 Winter Olympics in Salt Lake City.

NIETO uses TCP to attach to an NCID Server to retrieve and display Telephony Caller ID information on an attached LCD display. NIETO also has a real time clock on the LCD display which is periodically synched to an NTP time server using UDP NTP packets on the internet. User interface consists of 4 pushbuttons to page through a Caller ID log and to clear the log or reconnect to the Server.

NCID is an open source Caller ID server available on [NCID.sourceforge.net](http://NCID.sourceforge.net). It gathers Caller ID info from a phone line via a Caller ID modem for conventional analog telephony or via a SIP gateway for VOIP telephony and stores it in a log file for client retrieval. NIETO is a client that will attach to the NCID server, retrieve and store up to the last 80 entries in the log file and display the latest entry on the LCD. As an IP client, NIETO can attach to an NCID server without geographic bounds, retrieving Caller ID data across the internet. As new calls arrive, the latest will always be displayed.

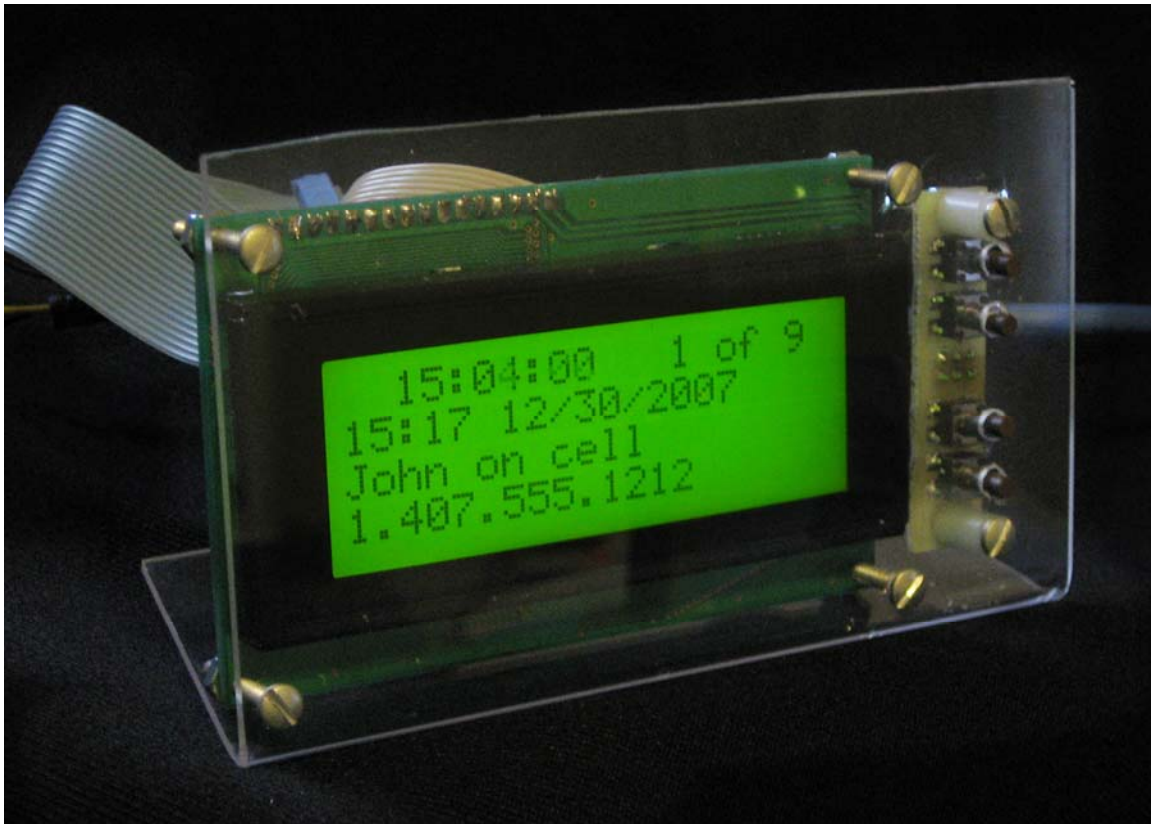
Pressing the Forward or Backward pushbutton will step forward or backward through the caller log. When the clear pushbutton is pressed, the local storage of the log file is cleared. NIETO is normally continuously connected to the Server. Pressing the Connect pushbutton will close the connection and re-open a TCP connection to the Server.

A timer interrupt performs timekeeping, updating the time variable once per second. The variable is the number of seconds elapsed since 1-1-1900. Simple math routines parse this number into ASCII hours, minutes and seconds for LCD display. Periodically an NTP packet is sent to an Internet Time Server to keep NIETO in synch with a time standard. The periodic rate is programmable and is normally set to once per day.

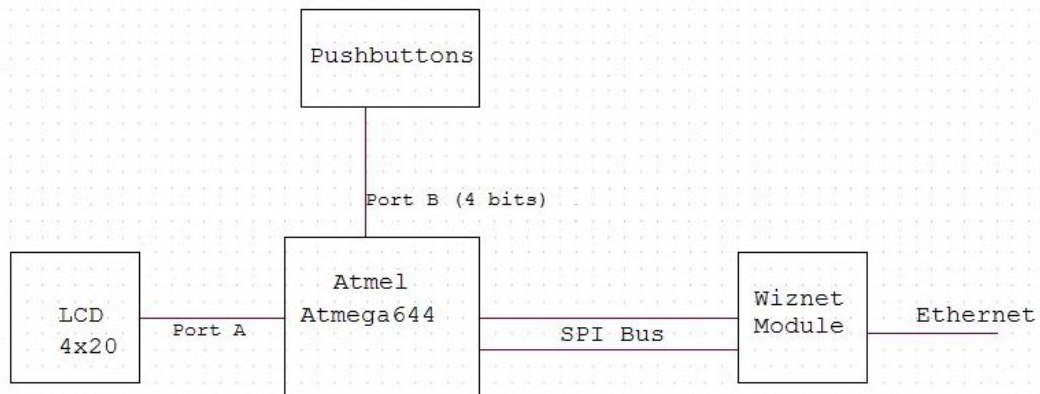
NIETO hardware consists of an Atmel Atmega644, a Wiznet W5100, a 4x20 LCD display and 4 pushbuttons. Communication between the Atmega644 and the Wiznet W5100 module is via the 4 wire SPI bus. An onboard regulator takes a 5 Volt input voltage and produces 3.3 Volts for the Atmega644 CPU and the W5100 Wiznet modules.

Not included in this submission but under development is a web server for easy project configuration. Currently all addresses are compiled in the source code. Future revisions will include the WebServer onboard the Atmega using TCP port 80 of the W5100.

This software was designed, compiled and developed using WinAVR.v2007\_0525

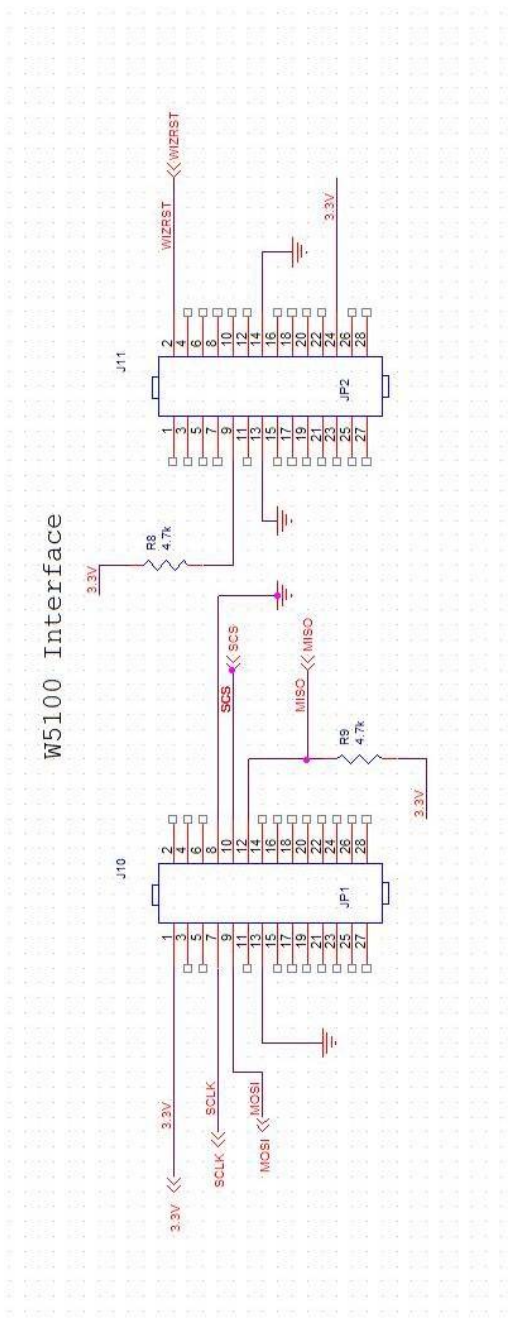


**Photo 1** with most recent call on LCD display. Four pushbuttons can be seen at the right of the display. The top line contains the real time NTP clock. The ribbon cable connects the display to the processor board behind the LCD. Reg No. 001171



**Block Diagram NIETO architecture** Reg No. 001171





```

/*****
* ntp_request
*   Open UDP to an NTP port and send an NTP request packet.
*
*   Return length for success, 0 for fail
*****/
unsigned char ntp_request(void)
{
    unsigned char i;
    unsigned char *ptr;
    unsigned char len;
    unsigned char running = 1;

    if (socket(NTP_SOCKET, Sn_MR_UDP, NTP_SOURCE_PORT, 0))
    {
        // fill out the NTP data
        ptr = &tx_buff[0];

        *ptr++ = 0xe3; // ntp hdr flgs
        *ptr++ = 0x0; // canned ntp req next 8 bytes
        *ptr++ = 0x04;
        *ptr++ = 0xfa;
        *ptr++ = 0x0;
        *ptr++ = 0x1;
        *ptr++ = 0x0;
        *ptr++ = 0x1;
        *ptr++ = 0x0;
        *ptr++ = 0x0;
        for (i=0;i<40;i++) *ptr++=0; // zero rest of data
        // short delay after socket open before sending data
        tmr0_gp=100;
        while (tmr0_gp);
        // send the buffer to target IP time_ip
        sendto(NTP_SOCKET, tx_buff, 0x30, time_ip, NTP_DEST_PORT);
    }

    tmr0.secs1=5; // timeout time for NTP reply. Abandon if timer has been reached without an NTP response

    do {
        if (!tmr0.secs1) running = 0;
        len = getSn_RX_RSR(NTP_SOCKET);
        if (len)
        {
            running = 0;
            recvfrom(NTP_SOCKET, rx_buff, 10, time_ip, NTP_DEST_PORT);
        }
    }while (running == 1);

    return len;
}

```

**Listing 1. NTP Request** Reg No. 001171

```

/*****
* ntp_parse
*   Parse the returned NTP frame and extract time data.
*
*****/
void ntp_parse(void)
{
    unsigned long *rawsecs;

    // point to long val
    rawsecs = &tmr0.unix_secs;

    // extract the transmit time stamp from the ntp server packet and make into unsigned long
    *rawsecs=((uint32)rx_buff[0x28]<<24)|((uint32)rx_buff[0x29]<<16)|((uint32)rx_buff[0x2a]<<8)|((uint32)rx_buff[0x2b]);

    // adjust to seconds since 1-1-1970 (unix time)
    tmr0.unix_secs -= SECS_NTP_OFFSET;

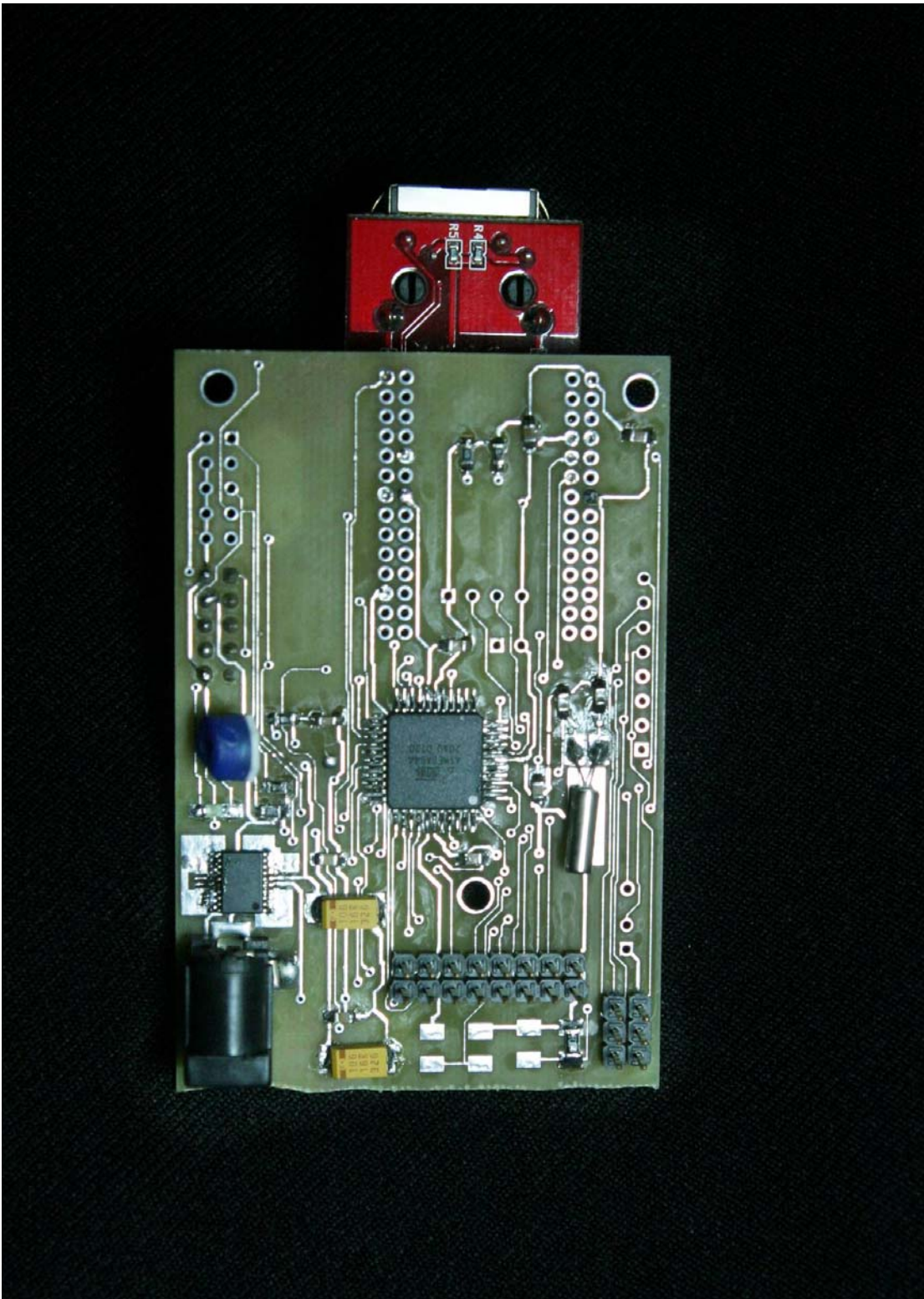
    // offset from UTC for local time
    tmr0.unix_secs -= (UTC_ADJ_HRS*3600);

    // reset the ntp refresh counter since this function is usually called after
    // a new NTP request
    tmr0.ntp_refresh = (unsigned long)60*UTC_REFRESH_MINS;

}

```

**Listing 2** NTP Parse. Extract the returned time from buffer and convert to seconds since 1-1-1900, and adjust for time zone from UTC time Reg No. 001171



**Photo 2.** The Atmega644 and Wiznet. The header pins connect to the LCD and pushbuttons. Reg No. 001171