Abstract

Open Source Software development with engineers located al over the world seems to work well. Tools such as CVS, SVN and GIT mean that a large distributed team of engineers can work on a piece of software collaboratively and produce very impressive results. Of course this works because every engineer has the hardware (ie a PC) on which to test their work.

The same cannot be said for embedded software projects. Generally every engineer needs a set of the embedded hardware so that they can test and debug their work. Consequently collaboratively efforts on embedded software projects are more restricted.

This project is designed to help get around this limitation by providing a system to remotely access an embedded development system. One of the more common means to access such a system is through a serial port connection. This project provides the 'server' end of such connectivity (accessed via a virtual serial port application at the client end). However often the final missing thing of remote access is the ability to press the reset button and turn on and off the power switch - the 'remote finger' also provides this last feature!

Features:

- Uses the WIZnet W7100 device (Internet MCU integrating HW TCP/IP core with 8051 processor)
- TCP/IP to serial port server to provide remote access to an RS232 port.
- Two controlled power points (2A 250V max)
- One controlled relay (SPDT)
- A web page server through which the RS232 port speed, power points and relays can be controlled
- Webpage includes an 'in use by' field to allow control of access to the system (allow multiple users to share a single resource)
- A Serial Port server to provide remote RS232 support
- Remote control through command line using the 'wget' utility.
- An LCD that shows the current status of the power outputs and relay

Operation

The Remote Finger provides an HTML server through which it is controlled. It returns this page:

This page provides the following features:

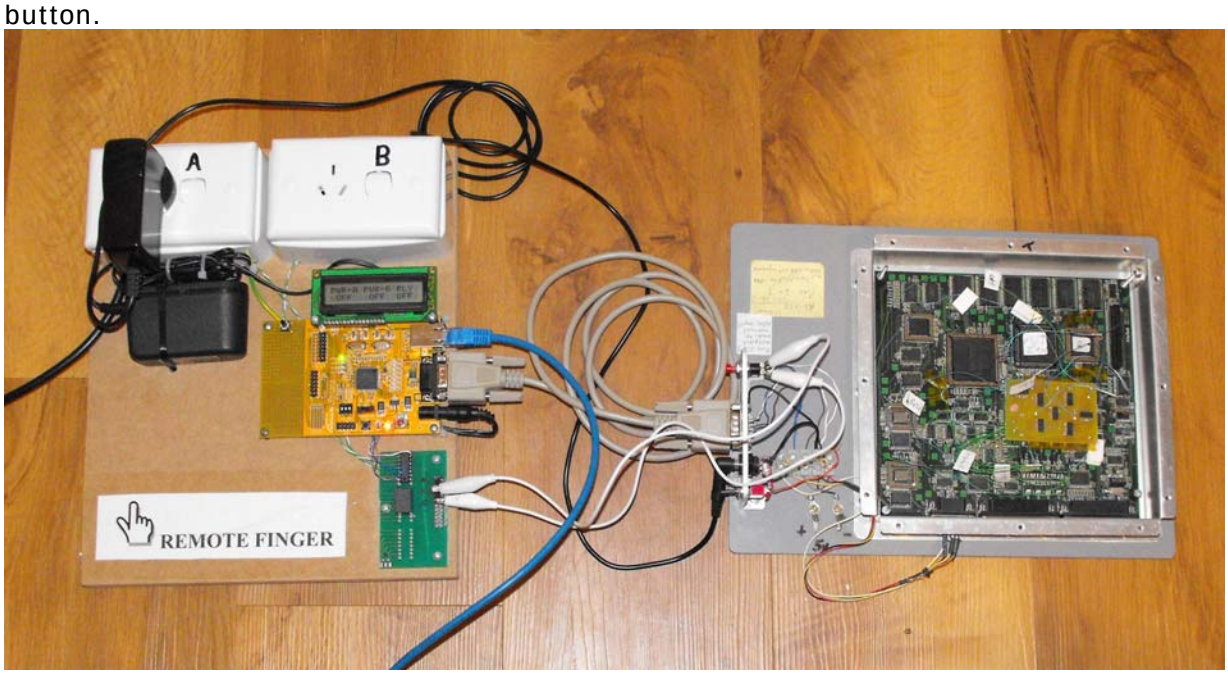
- Displays the current status of the Remote Finger relays and RS232 port speed.
- Provides an indication of the current user on the system (to manage multiple user access).
- Provides links to change the relays and serial port speed.
- Includes an embedded image

Remote Finger control is also possible via a PC command line using a utility such as ‘wget’.

The Remote Finger serial port can be access using a PC application such as the HW Group’s ‘VSP3’ (Virtual Serial Port 3). This will make the Remote Finger serial port appear as if it is a serial port directly connected to the client PC.

Photographs

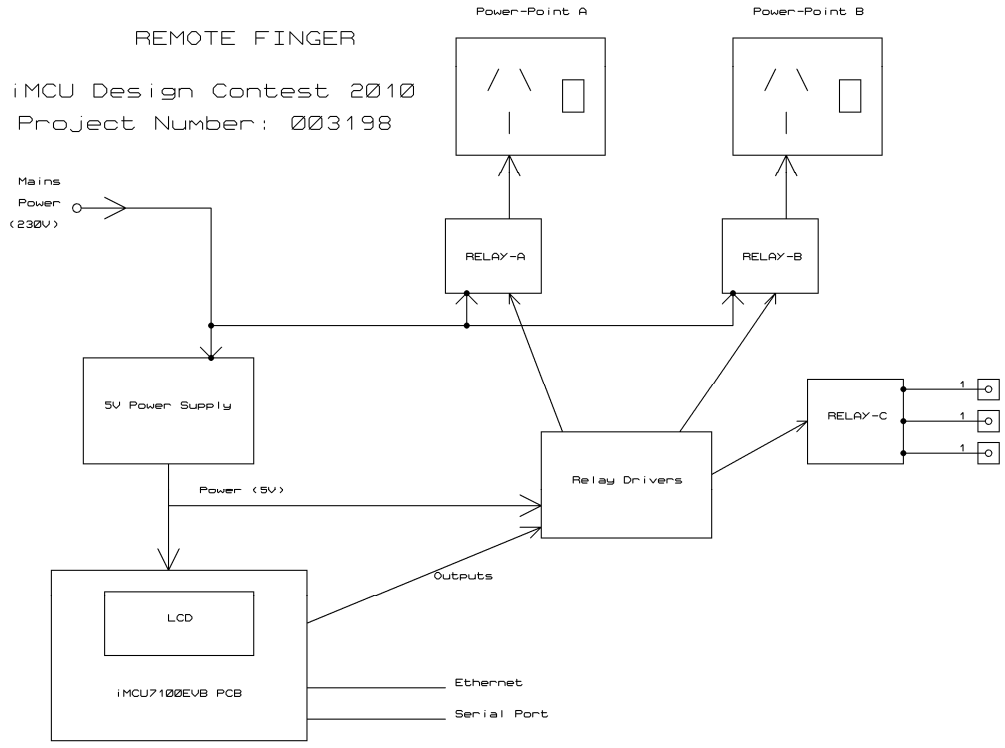
The Remote Finger (left) in use, connected to a system under development (right). Note the clip-leads that connect the relay contacts to the development system RESET button.



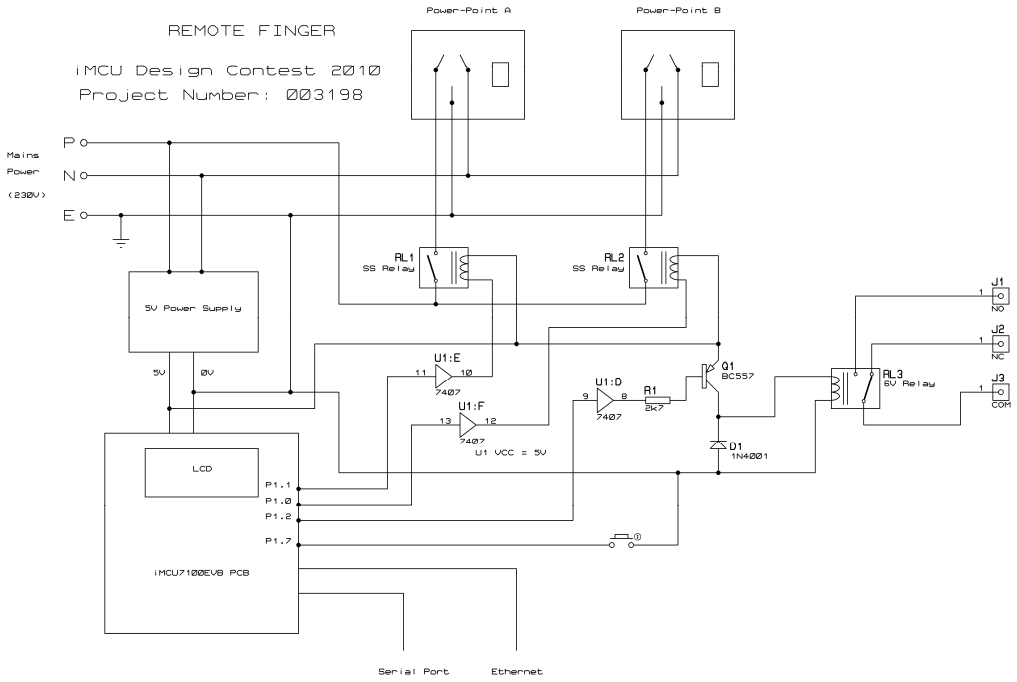
Block Diagram

The hardware is based on the iMCU7100EVB evaluation PCB. This is mounted onto an MDF base board along with the other hardware. Wired to the PCB are the following:

- Two power-points with associated solid-state power relays (suitable enclosed for safety)
- One relay (6V coil) providing an isolated SPDT switch
- Relay driver IC and transistor circuit
- 5V power supply for the evaluation PCB



Schematic


REMOTE FINGER
iMCU Design Contest 2010
Project Number: 003198
Power-Point A
Power-Point B
Mains Power (230V)
P
N
E
RL1
SS Relay
RL2
SS Relay
5V Power Supply
5V
0V
U1:E
7407
U1:F
7407
U1:D
7407
U1 VCC = 5V
R1
2k7
Q1
BC557
D1
1N4001
RL3
5V Relay
J1
NO
J2
NC
J3
COM
LCD
P1.1
P1.0
P1.2
P1.7
iMCU7100EVB PCB
Serial Port
Ethernet

Software Example

From the HTML server (some sections removed):

```
/*---------------------------------------------------------------------------*/
/* Process a line of data from the socket
 */
static void html_process_line( void )
{
    uint8 xdata *pLine;
    int     ii,jj;

    if ( html_in_count==0 )
    {   // an empty line
        // this indicates the end of the HTML request,
         // and hence the time to return the html page
        htmlserver_generatepage();
        sprintf( sprintf_buffer,
                    "HTTP/1.1 200 OK\n"
                    "Content-Length: %d\n",
                        strlen( html_page ) );
        send( HTML_SOCKET, sprintf_buffer, strlen(sprintf_buffer) );
        sprintf( sprintf_buffer,
                    "Connection: close\n"
                    "Content-Type: text/html\n"
                    "\n" );
        // send the header
        send( HTML_SOCKET, sprintf_buffer, strlen(sprintf_buffer) );
        // return the page section by section
        ii = strlen(html_page);
        pLine = html_page;
        while ( ii>512 )
        {
            send ( HTML_SOCKET, pLine, 512 );
            pLine += 512;
            ii -= 512;
        }
        send( HTML_SOCKET, pLine, ii );
        // disconnect the socket
        disconnect( HTML_SOCKET );
        return;
    }
    // parse the line
    pLine = html_in_buf;
    while ( *pLine==' ')
        pLine++;
    // look for the 'GET'
    if ( (pLine[0]=='G') && (pLine[1]=='E') && (pLine[2]=='T') && (pLine[3]==' ') )
    {   // a 'GET' line
        pLine += 3;
        ii = 0;
        // strip out spaces
        while ( *pLine==' ')
            pLine++;
        // copy out the page
        while ( *pLine>' ' )
        {
            requested_page[ii] = *pLine++;
            if ( ii<HTML_PAGENAME_SIZE )
                ii++;
            requested_page[ii] = 0;
        }
        // process this!
        pLine = requested_page;
        if ( *pLine=='/' )
            pLine++;
        // User control
        if ( memcmp(pLine,"setuser?user=",13)==0 )
        {   // set the user
            // translate '+' to a space
            // translate %nn to hex char value
            ii = 13;    // start point of input
            jj = 0;     // destination index
            username[jj] = 0;
            while( pLine[ii]!=0 )
            {
                // check size
                if ( jj>=USERNAME_SIZE )
                    break;
                // translate + to space
```

```
                if ( pLine[ii]=='+' )
                {
                    username[jj] = ' ';
                    jj++;
                    username[jj] = 0;
                    ii++;
                    continue;
                }
                // translate %nn value
                if ( (pLine[ii]=='%') && ishex(pLine[ii+1]) && ishex(pLine[ii+2]) )
                {
                    username[jj] = hexvalue(pLine[ii+1])*0x10 + hexvalue(pLine[ii+2]);
                    jj++;
                    username[jj] = 0;
                    ii+=3;
                    continue;
                }
                // default
                username[jj] = pLine[ii];
                jj++;
                username[jj] = 0;
                ii++;
            }
        }
        if ( memcmp(pLine,"resetuser?",10)==0 )
        {   // reset the user
            username[0] = 0;
        }
        // Relay control
        if ( strcmp(pLine,"RelayAOn")==0 )
            relayA = true;
        if ( strcmp(pLine,"RelayAOff")==0 )
            relayA = false;
…
        // Comport speed control
        if ( strcmp(pLine,"port1200")==0 )
        {
            sio_set_baud(B1200);
            baudrate = 1200;
        }
        if ( strcmp(pLine,"port2400")==0 )
        {
            sio_set_baud(B2400);
            baudrate = 2400;
        }
…
        // set up outputs
        if ( relayA )
            RLY_ON(RLY_A);
        else
            RLY_OFF(RLY_A);
        if ( relayB )
            RLY_ON(RLY_B);
        else
            RLY_OFF(RLY_B);
        if ( relayC )
            RLY_ON(RLY_C);
        else
            RLY_OFF(RLY_C);
        // display LCD
        lcd_set_text (0, "PWR-A PWR-B RLY ");
        if ( relayA )
            strcpy( lcdstr, " ON   " );
        else
            strcpy( lcdstr, " OFF  " );
        if ( relayB )
            strcat( lcdstr, " ON  " );
        else
            strcat( lcdstr, " OFF " );
        if ( relayC )
            strcat( lcdstr, " ON  " );
        else
            strcat( lcdstr, " OFF " );
        lcd_set_text (1, lcdstr );
    }
}
```