243-848-92 – Computer Project

Progress Report #3

Calvin Ouellet-Ference
A711643
Mr. Markou
Submitted on the 22nd of April 2010

1. Objectives

The objectives of of the last four weeks were to get the minimal system to communicate with my laptop and to get a mechanism working for the water level detection.

2. Progress

2.1 Ethernet Controller

The Ethernet controller is working but not quite. It can send messages through the network without any problems. However when it comes to receiving, it only reads 00h. So to be able to do something, the PC application will send X number of packets to activate a certain function. Overall, this module is completed.

2.2 Coffee Machine

For the coffee machine module, the switching circuit with the relay was attached to the power cable and is fully functional. To determine the water level, an IR sensor/receiver is being used (the SHARP GP2D120). It has an effective range of 4cm to 30cm and outputs an analog signal which is then fed into an AD0804. The sensor is fully functional and interfaced with the minimal system. Some difficulties arose while trying to determine the water level as the sensor can not determine the water level without a float. So a float made was built using an plastic spoon folded in half wrapped in electrical tape and aluminum foil. Which seems to work quite well. However while doing some readings to try and find what water level is equal to what value, I've ran into some problems. The most apparent one is the following: I'd measure the level without water with a test program and the IR sensor, then I added two cups and measure again, then add up to three cups and henceforth. However, when taking the readings while emptying the water, the values were not matching up which is unusual. I theorized on two possibilities. One; the steam that is rising from the water tank is causing reading issues. Two; the steam rising has melted the glue which fixes the IR sensor on the lid of the coffee machine. To fix this problem, I will fix the IR sensor with bolts instead of glue as I've successfully determined the melting point of epoxy which is about 90°C. Once that is bolted, I will redo the measurements and determine if the problem was option two. If not I'll have to device a way to get option one to read properly.

2.3 Computer Application

Some issues with the timeout loop seem to have hold up most of my time with this module. After going through all sorts of documentation, setting the socket options to include a timeout seem to have worked perfectly. This seems odd to me as the man pages state that the timeout function under Linux is not available. With the timeout working, I can now successfully have a reliable UDP connection where the application sends an UDP packet and waits for a reply from the minimal system. Also, I had to manually enter the MAC address of the minimal system into the computer using the arp -s command. I will try to incorporate this straight into the program. I also need to find a way so that you do not need to use root privileges to access the network card and the arp command.

2.4 Power Supply

The power supply was built with a +5V rail and a +12V rail. It seems to be fully functional but I still need to do a burn-in test to see if it last more than 5 minutes. A fan might be necessary just to cool down the 7805 regulator as it's dissipating 5.832W of power. This was calculated with the following:

$$P_D=(V_I-V_O)I_L+V_I I_G$$
$$I_I=I_L+I_G$$
$$I_G=I_I-I_L$$
$$P_D=(V_I-V_O)I_L+V_I(I_I-I_L)$$

$$P_D=(16\text{V}-5\text{V})500\text{mA}+16\text{V}\,(520.832\text{mA}-500\text{mA})$$
$$P_D=(11\text{V})500\text{mA}+16\text{V}\,20.832\text{mA}$$
$$P_D=5.5\text{W}+0.332\text{W}$$
$$P_D=5.832\text{W}$$

3. Circuits Schematics

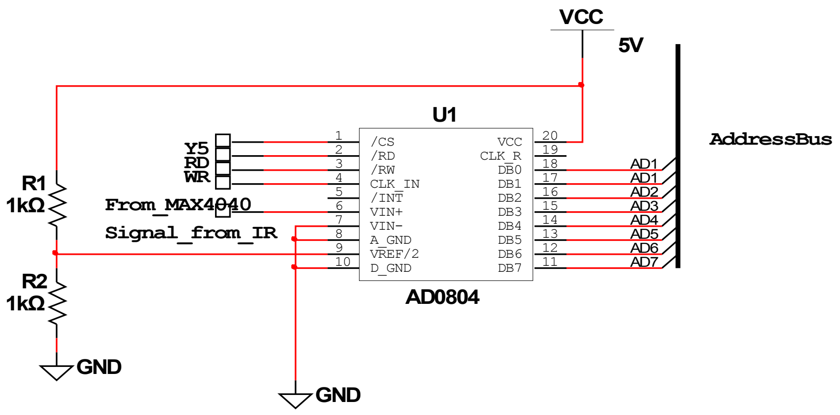

Figure 3.1 – Analog to Digital Converter Schematic

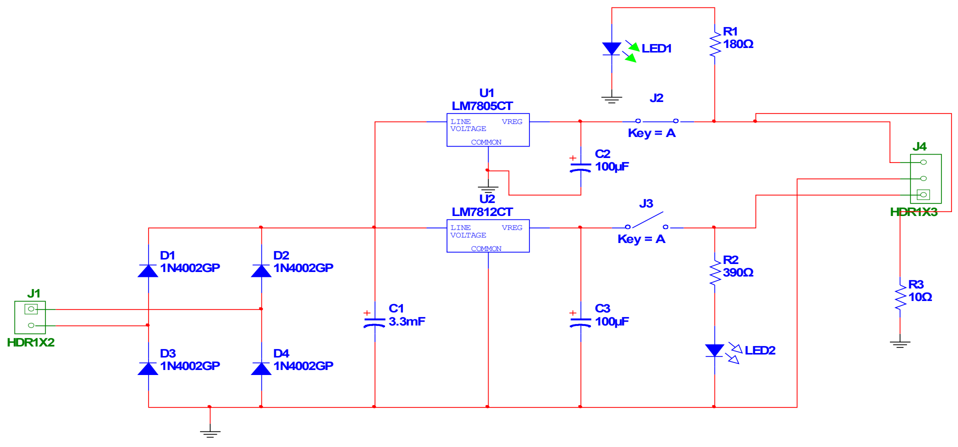

Figure 3.2 – Power Supply Unit Schematic

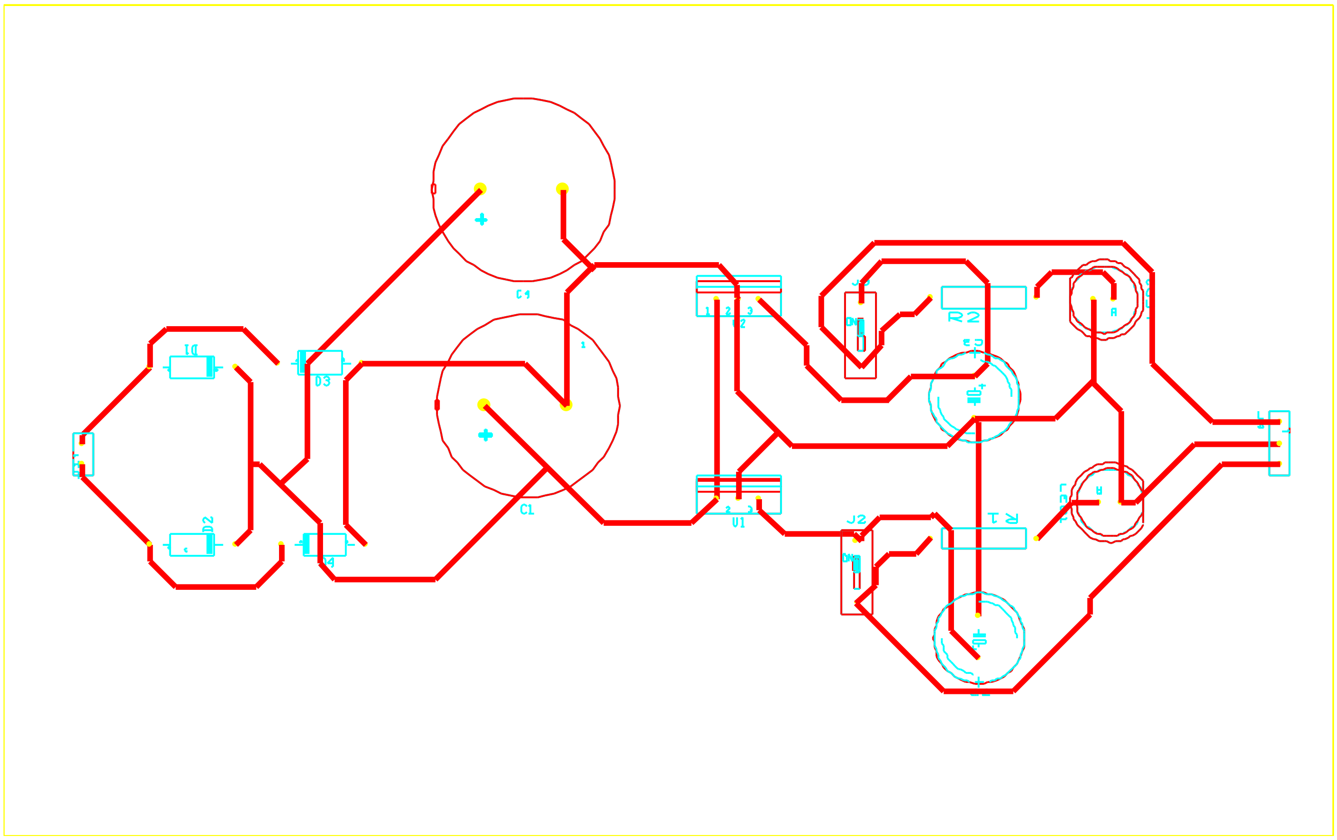
Figure 3.3 – Power Supply Unit PCB Layout

4. Code

4.1 Sensor Test Program

```
.model tiny
.code
; external commands (m88io.obj) preprocessor
EXTRN       newline:NEAR, outbyte:NEAR, outword:NEAR, getc:NEAR, outc:NEAR,
outstr:NEAR

; I/O preprocessor
LED                   equu           0F00h
; PIC preprocessor
PIC                   equ            400h
ICW1        equ          PIC
ICW2        equ          PIC+1
ICW4        equ          PIC+1
OCW1        equ          PIC+1
ICW1B       equ          00010011b   ;edge trig
ICW2B       equ          01000000b   ;vec. no. 40h = 100h
ICW4B       equ          00000011b
OCW1B       equ          11111110b
FREQ        equ          2400

ADCO equ    800h
```

```
        org     0800h
main    proc
init:
                        cli
                        mov             ax,0
        mov     al,10010000b    ;configuration word for the 8255
                                ;both group A and B = mode 0
                                ;port A = input
                                ;port B = output
                                ;port C = output
        mov     dx,PPICTL
        out     dx,al           ;send the configuration word
                        mov             dx,0
                        mov             cx,0

init_pic:
                        lea     di,rtc
                        mov     ds:[100h],di
                        mov     ax,0
                        mov     ds:[102h],ax

                        mov     dx,ICW1
                        mov     al,ICW1B
                        out     dx,al

                        mov     dx,ICW2
                        mov     al,ICW2B
                        out     dx,al

                        mov     dx,ICW4
                        mov     al,ICW4B
                        out     dx,al

                        mov     dx,OCW1
                        mov     al,OCW1B
                        out     dx,al

here:   mov             time,0
                        mov             dx,ADCO
                        mov             al,0ffh
                        out             dx,al
; wa1t:
                        ; sti
                        ; cmp           time,1
                        ; jbe           wa1t
                        ; cli
                        mov             dx,ADCO
                        in              al,dx
                        call    outbyte
```

```
                mov             al," "
                call    outc
                 jmp            here


rtc:
                inc             tick
                cmp             tick,FREQ
                jbe             idone
                mov             tick,0
                inc             time
idone:
                iret

tick            dw      0
time            db  0

main    endp
                end
```

Essentially this program just reads the Analog to Digital converter and outputs the hexadecimal value on the screen in an endless loop. When the program is running, we can see the change in distance just by moving the sensor around.

4.2 Minimal System Protocol

```
.model tiny
.code
; external commands (m88io.obj) preprocessor
EXTRN           newline:NEAR, outbyte:NEAR, outword:NEAR, getc:NEAR, outc:NEAR,
outstr:NEAR
;eth0 preprocessor
ETH0            equ     0A00h
RXTX0L                  equ             ETH0+00h
RXTX0H                  equ             ETH0+01h
RXTX1L                  equ             ETH0+02h
RXTX1H                  equ             ETH0+03h
TXCMDL                  equ             ETH0+04h
TXCMDH                  equ             ETH0+05h
TXLENGTHL               equ             ETH0+06h
TXLENGTHH               equ             ETH0+07H
ISQL            equ             ETH0+08h
ISQH            equ             ETH0+09H
PPPL            equ             ETH0+0AH
PPPH            equ             ETH0+0BH
PPD0L           equ             ETH0+0CH
PPD0H                   equ             ETH0+0DH
PPD1L           equ             ETH0+0EH
```

```
PPD1h           equ             ETH0+0FH
; I/O preprocessor
LED                     equ             0E00h
PPI                     equ             0200h
PORTA           equ     PPI+0
PORTB           equ     PPI+1
PORTC           equ     PPI+2
PPICTL          equ     PPI+3
; PIC preprocessor
PIC                     equ             400h
ICW1            equ             PIC
ICW2            equ             PIC+1
ICW4            equ             PIC+1
OCW1            equ             PIC+1
ICW1B           equ             00010011b   ;edge trig
ICW2B           equ             01000000b   ;vec. no. 40h = 100h
ICW4B           equ             00000011b
OCW1B           equ             11111110b
FREQ            equ             2400
; ADC preproc
ADCON                   equ             0800h

        org     0800h
main    proc
init_ppi:
                cli
                mov             ax,0
        mov     al,10010000b    ;configuration word for the 8255
                        ;both group A and B = mode 0
                        ;port A = input
                        ;port B = output
                        ;port C = output
        mov     dx,PPICTL
        out     dx,al           ;send the configuration word
                mov             dx,0
                mov             cx,0

                mov             al,0ffh
                mov             dx,PORTC
                out             dx,al
init_pic:
                lea     di,rtc
                mov     ds:[100h],di
                mov     ax,0
                mov     ds:[102h],ax

                mov     dx,ICW1
                mov     al,ICW1B
                out     dx,al
                mov     dx,ICW2
                mov     al,ICW2B
                out     dx,al

                mov     dx,ICW4
                mov     al,ICW4B
                out     dx,al

                mov     dx,OCW1
                mov     al,OCW1B
                out     dx,al

reset_wait:                                     ; Just to give eth0 enough time to
                inc             cx              ; init internally
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                cmp             cx,07fffh
                jbe             reset_wait
                mov             dx,LED
                mov             al,01h
                out             dx,al
eth0_init:
;MAC_INIT
                mov             dx,PPPL
                mov             al,12h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov             al,0D3h
                out             dx,al
                mov             dx,PPD0H
                mov             al,00h
                out             dx,al
                mov             dx,PPPL
                mov             al,58h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                mov         al,000h
                out         dx,al
                mov         dx,PPD0H
                mov         al,6Fh
                out         dx,al
                mov         dx,PPPL
                mov         al,5Ah
                out         dx,al
                mov         dx,PPPH
                mov         al,01h
                out         dx,al
                mov         dx,PPD0L
                mov         al,066h
                out         dx,al
                mov         dx,PPD0H
                mov         al,66h
                out         dx,al
                mov         dx,PPPL
                mov         al,5Ch
                out         dx,al
                mov         dx,PPPH
                mov         al,01h
                out         dx,al
                mov         dx,PPD0L
                mov         al,065h
                out         dx,al
                mov         dx,PPD0H
                mov         al,65h
                out         dx,al
;TestCTL
                mov         dx,PPPL
                mov         al,18h
                out         dx,al
                mov         dx,PPPH
                mov         al,01h
                out         dx,al
                mov         dx,PPD0L
                mov         al,00h ;10011001b
                out         dx,al
                mov         dx,PPD0H
                mov         al,00h ;01000000b
                out         dx,al
; LineCTL
                mov         dx,PPPL
                mov         al,12h
                out         dx,al
                mov         dx,PPPH
                mov         al,01h
                out         dx,al
```

```
		mov		dx,PPD0L
		mov		al,0D0h
		out		dx,al
		mov		dx,PPD0H
		mov		al,00000000b
		out		dx,al

		mov		dx,PPPL
		mov		al,04h
		out		dx,al
		mov		dx,PPPH
		mov		al,01h
		out		dx,al
		mov		dx,PPD0L
		mov		al,01000000b
		out		dx,al
		mov		dx,PPD0H
		mov		al,00111001b
		out		dx,al

		call	display_mac
poll:
		mov		dx,PPPL
		mov		al,24h
		out		dx,al
		mov		dx,PPPH
		mov		al,01h
		out		dx,al
		mov		dx,PPD0L
		in		al,dx
		mov		tmp,al
		mov		dx,PPD0H
		in		al,dx
		mov		ah,al
		mov		al,tmp
		mov		wtmp,ax

		and		ax,0ff0fh
		cmp		ax,2304h
		jne		poll
		;mov	ax,wtmp
		;call	outword
	;	call	newline
		;call	send
		call	recv
		cmp		p_cnt,6
		je		water_level
		cmp		p_cnt,5
		je		brew
```

```
                cmp             p_cnt,4
                je              stop
                jmp             poll

water_level:
                mov             dx,ADCON
                in              al,dx
                cmp             al,00h          ;5 Cups
                jae             cups_5
                cmp             al,00h          ;4 Cups
                jae             cups_4
                cmp             al,00h          ;2 Cups
                jae             cups_2
cups_0:
                mov             cups_flag,0
                jmp             cups_display
cups_2:
                mov             cups_flag,2
                jmp             cups_display
cups_4:
                mov             cups_flag,4
                jmp             cups_display
cups_5:
                mov             cups_flag,5
cups_display:
                lea             di,cups_msg
                call    outstr
                call    newline
                mov             al,cups_flag
                call    outbyte
                jmp             poll

brew:
                lea             di,brewing
                call    outstr
                call    newline
                mov             al,00
                mov             dx,PORTC
                out             dx,al
                jmp             poll
stop:
                lea             di,stopping
                call    outstr
                call    newline
                mov             al,0ffh
                mov             dx,PORTC
                out             dx,al
                jmp             poll
```

```
rtc:
                inc             tick
                cmp             tick,FREQ
                jbe             idone
                mov             tick,0
                inc             time
idone:
                iret
main    endp


recv    proc
                sti

                call    newline
                mov             al,"!"
                call    outc

                mov             p_cnt,1
                mov             data,0A0h
                mov             data+1,0A0h
                call    send
recv_poll:
                cmp             time,5 ; 10 Seconds
                jae             shi

                mov             dx,PPPL
                mov             al,24h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al

                mov             dx,PPD0L
                in              al,dx
                mov             tmp,al
                mov             dx,PPD0H
                in              al,dx
                mov             ah,al
                mov             al,tmp          ;data

                and             ax,0f0ffh
                cmp             ax,2044h
                jne             recv_poll

                inc             p_cnt

                mov             data,0A0h
                mov             data+1,0A0h
                call    send            ;ACK
                mov             time,0          ;Reset the timer
                jmp             recv_poll
shi:                                            ;; HERE!! Do not know byt I think the problem may lie in
shi
                cli
                call    newline
                lea             di,recv_msg
                call    outstr
                mov             al,p_cnt
                call    outbyte
                call    newline
                mov             time,0
                ret
recv    endp


display_mac     proc
                lea             di, mac
                mov             dx,PPPL
                mov             al,58h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                in              al,dx
                mov             [di],al
                inc             di
                mov             dx,PPD0H
                in              al,dx
                mov             [di],al
                inc             di
                mov             dx,PPPL
                mov             al,5Ah
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
                mov             dx,PPD0L
                in              al,dx
                mov             [di],al
                inc             di
                mov             dx,PPD0H
                in              al,dx
                mov             [di],al
                inc             di
                mov             dx,PPPL
                mov             al,5Ch
```

```
            out         dx,al
            mov         dx,PPPH
            mov         al,01h
            out         dx,al
            mov         dx,PPD0L
            in          al,dx
            mov         [di],al
            inc         di
            mov         dx,PPD0H
            in          al,dx
            mov         [di],al
            inc         di
            lea         di, mesg_mac
            call    outstr
            lea         di,mac
            mov         al,[di]
            call    outbyte
            mov         al,3Ah
            call    outc
            inc         di
            mov         al,[di]
            call    outbyte
            mov         al,3Ah
            call    outc
            inc         di
            mov         al,[di]
            call    outbyte
            mov         al,3Ah
            call    outc
            inc         di
            mov         al,[di]
            call    outbyte
            mov         al,3Ah
            call    outc
            inc         di
            mov         al,[di]
            call    outbyte
            mov         al,3Ah
            call    outc
            inc         di
            mov         al,[di]
            call    outbyte
            ret
display_mac endp

send    proc
;setting up the TxCMD
            mov         dx,TXCMDL
            mov         al,0C0h
```

```
                out             dx,al
                mov             dx,TXCMDH
                mov             al,00h
                out             dx,al
;setting up the TxLength
                mov             dx,TXLENGTHL
                mov             al,78h ;2Bh                 ;[LENGTH!!!!!]
                out             dx,al
                mov             dx,TXLENGTHH
                mov             al,00h
                out             dx,al
;Packet Page Pointer Set-up
PPP:
                mov             dx,PPPL
                mov             al,38h
                out             dx,al
                mov             dx,PPPH
                mov             al,01h
                out             dx,al
;Reading the Packet Page Pointer Data

                mov             dx,PPD0H
                in              al,dx
                and             al,01h
                cmp             al,01h
                jne             PPP
        ;       lea             di,udp_hdr
                mov             cl,0
;start moving data
tx_data:
                ;destination MAC
                mov             dx,RXTX0L
                mov             al,00h
                out             dx,al
                mov             dx,RXTX0H
                mov             al,26h
                out             dx,al
                mov             dx,RXTX0L
                mov             al,2dh
                out             dx,al
                mov             dx,RXTX0H
                mov             al,7ch
                out             dx,al
                mov             dx,RXTX0L
                mov             al,073h
                out             dx,al
                mov             dx,RXTX0H
                mov             al,0b5h
                out             dx,al
```

```
;Source MAC
mov         dx,RXTX0L
mov         al,43h
out         dx,al
mov         dx,RXTX0H
mov         al,6fh
out         dx,al
mov         dx,RXTX0L
mov         al,66h
out         dx,al
mov         dx,RXTX0H
mov         al,66h
out         dx,al
mov         dx,RXTX0L
mov         al,65h
out         dx,al
mov         dx,RXTX0H
mov         al,65h
out         dx,al
;type
mov         dx,RXTX0L
mov         al,08h
out         dx,al
mov         dx,RXTX0H
mov         al,00h
out         dx,al
;ip hdr
;version, header length
mov         dx,RXTX0L
mov         al,45h
out         dx,al
;services
mov         dx,RXTX0H
mov         al,00h
out         dx,al
mov         dx,RXTX0L
mov         al,00h
out         dx,al
;total length
mov         dx,RXTX0H
mov         al,14h
out         dx,al
;ID
mov         dx,RXTX0L
mov         al,01h
out         dx,al
mov         dx,RXTX0H
mov         al,40h
out         dx,al
```

```
;Flags
mov             dx,RXTX0L
mov             al,00h
out             dx,al
;Fragment Offset
mov             dx,RXTX0H
mov             al,00h
out             dx,al
;TTL
mov             dx,RXTX0L
mov             al,05h
out             dx,al
;protocl
mov             dx,RXTX0H
mov             al,11h
out             dx,al
;Checksum
mov             dx,RXTX0L
mov             al,031h
out             dx,al
mov             dx,RXTX0H
mov             al,0b8h
out             dx,al
;Source Address
mov             dx,RXTX0L
mov             al,0c0h
out             dx,al
mov             dx,RXTX0H
mov             al,0a8h
out             dx,al
mov             dx,RXTX0L
mov             al,00h
out             dx,al
mov             dx,RXTX0H
mov             al,0e1h
out             dx,al
;Destination Address
mov             dx,RXTX0L
mov             al,0c0h
out             dx,al
mov             dx,RXTX0H
mov             al,0a8h
out             dx,al
mov             dx,RXTX0L
mov             al,00h
out             dx,al
mov             dx,RXTX0H
mov             al,0b0h
out             dx,al
```

```
                ;udp hdr
                ;souce port
                mov             dx,RXTX0L
                mov             al,26h
                out             dx,al
                mov             dx,RXTX0H
                mov             al,17h
                out             dx,al
                ;destination port
                mov             dx,RXTX0L
                mov             al,26h
                out             dx,al
                mov             dx,RXTX0H
                mov             al,17h
                out             dx,al
                ;length
                mov             dx,RXTX0L
                mov             al,00h
                out             dx,al
                mov             dx,RXTX0H
                mov             al,52h
                out             dx,al
                ;chksum
                mov             dx,RXTX0L
                mov             al,chk_sum
                out             dx,al
                mov             dx,RXTX0H
                mov             al,chk_sum+1
                out             dx,al
                ;data
                mov             dx,RXTX0L
                mov             al,data
                out             dx,al
                mov             dx,RXTX0H
                mov             al,data+1
                out             dx,al
                mov             cl,0
pad:
        ;       ;padding
                mov             dx,RXTX0L
                mov             al,0aah
                out             dx,al
                mov             dx,RXTX0H
                mov             al,0bbh
                out             dx,al
                inc             cl
                cmp             cl,23h
                jbe             pad
                mov             dx,RXTX0L
                mov         al,0aeh
                out         dx,al
                mov         dx,RXTX0H
                mov         al,058h
                out         dx,al
                mov         dx,RXTX0L
                mov         al,0cdh
                out         dx,al
                mov         dx,RXTX0H
                mov         al,073h
                out         dx,al
                mov         flag,1
                ret
send    endp

; Messages
mesg_mac        db "The MAC Address of this System is: ",04
end_msg                 db "Press any key to send another packet... ",04
recv_msg        db "I've received: ",04
brewing                 db "Brewing...",04
stopping        db      "Stopping...",04
cups_msg        db      "Cups of water: ",04
; eth0 Data
mac                     db 00,00,00,00,00,00
data            db 15h,51h
chk_sum         db 00, 00

;RTC
tick            dw      0
time            db  0

;flags
flag            db      0
cups_flag       db      0
; Temporary Storage Area
tmp             db  00h
wtmp            dw  0000h
p_cnt           db      00h             ;ping counter
                end
```

This program is the same as the one in progress report #2. With the only difference of some code that checks the ADC to see what is the water level at. The Values have still not been set has they haven't been properly measured yet.

4.3 Routing Set-up Script

```
#!/bin/sh
sudo ifconfig eth0 192.168.0.176
arp -s 192.168.0.225 00:6F:66:66:65:65
```

The above script does the following: gets sudo privileges and sets up eth0 with an IP address, this is because I don't get a DHCP response while connected directly to my minimal system and when I reset the minimal system, the LINK gets cut and the IP address gets reset. The second line is to add a route. I'm using the arp command with the -s handle to set a new route. The first argument is the IP address of the new route and the second is the MAC address. I will incorporate the arp command in the final version of the program.

4.4 Brew

4.4.1 haxxor.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// This displays an error message before exiting
void fatal(char *message)
{
        char error[100];
        strcpy(error, "[ERROR] Fatal Error ");
        strncat(error, message, 80);
        perror(error);
        exit(1);
}

// checking errors for malloc, usefull wrapper function
void *Malloc(unsigned int size)
{
        void *ptr;
        ptr = malloc(size);
        if(ptr == NULL)
                fatal("in Malloc() while allocating memory");
        return ptr;
}

// Used for dumping memory in hex (similar to a sniffer output)
void mem_dump(const unsigned char *dump_buffer, const unsigned int length)
{
        unsigned char byte;
        unsigned int x, y;
        for(x=0;x<length;x++){
                byte = dump_buffer[x];
                printf("%02x ", dump_buffer[x]);
                if(((x%16)==15) || (x==length-1)) {
                  for(y=0;y<15-(x%16); y++)
                    printf("   ");
                  printf("| ");
                  for(y=(x-(x%16));y<=x; y++) {
                    byte = dump_buffer[y];
                    if((byte>31) && (byte<127))
                      printf("%c", byte);
                    else
                      printf(".");
                  }
                  printf("\n");
                }
        }
}
```

4.4.2 networking.h

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

// This is used to send all bytes pointed by ptr, returns 1 on success, 0 on failure
int Send(int sockfd, unsigned char *buffer)
{
        int sent, left;
        left = strlen(buffer);
        while(left > 0){
                sent = send(sockfd, buffer, left, 0);
                if(sent == -1)
                        return 0;
                left -= sent;
                buffer += sent;
        }
        return 1;
}

int recv_l(int sockfd, unsigned char *dest_buffer)
{
#define EOL "\r\n"
#define EOL_SIZE 2
        unsigned char *ptr;
        int eol_matched = 0;
        ptr = dest_buffer;
        while(recv(sockfd, ptr, 1, 0) == 1) {
                if(*ptr == EOL[eol_matched]) {
                        eol_matched++;
                        if(eol_matched == EOL_SIZE) {
                                *(ptr+1-EOL_SIZE) = '\0';
                                return strlen(dest_buffer);
                        }
                } else {
                        eol_matched = 0;
                }
```

```
                    ptr++;
            }
            return 0;
}
```

4.4.3 brew.c

```
#include "haxxor.h"
#include "networking.h”

#define DEBUG printf("[DEBUG]\n");
#define ADDR "192.168.0.225"

int ping(void) {

    int i, recv_length = 0, sockfd, n=0, m=0;
    u_char buffer[9000];
    struct timeval tv;

    tv.tv_sec = 2;          ;sets up the values for the timeout
    tv.tv_usec = 0;

    if((sockfd = socket(PF_INET, SOCK_RAW, IPPROTO_UDP)) == -1)
                    fatal("in socket");
    send_udp();

    for(;;){
      if(setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv))){     ;2 sec timeout
            printf("setsockopt error\n");
            exit(1);}
      if((recv_length = recv(sockfd, buffer, 8000, 0)) <= 0){
            printf("NADA ");
            break;}
      printf("Got a %i byte packet\n", recv_length);
      mem_dump(buffer, recv_length);
      return 0;
    }
    return 1;
}

int send_udp(void) { // Might want to replace the exit(1) with return 1...

  char buffer[256];
  int sockfd, length, n;
  struct sockaddr_in server;

  if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0){          ;Sets up an UDP port
    printf("Error while opening socket!\n");
    exit(1);}
  server.sin_family = AF_INET;                          ;IP Protocol
  inet_pton(AF_INET, ADDR, &server.sin_addr);
  server.sin_port = 9751;                               ;Port Number
  length = sizeof(struct sockaddr_in);

  bzero(buffer, 256);
  strcpy(buffer, "Brew me one!");

  n = sendto(sockfd, buffer, strlen(buffer), 0,(struct sockaddr*) &server, length);

  if(n<0){
    printf("Error while sending message!\n");
    exit(1);}

  return 0;
}


int main(int argc, char *argv[]) {

  int n, x;
  system("clear");
  printf("Press:\n(0) to stop the machine\n(1) to start the machine\nAnything else to quit\n\n>> ");
  scanf("%i", &x);

  if(x == 1){ // REPLACE WITH SWITCH?
    for(n=0;n<5;n++){
      if(ping())
          n--;
          //break;
    } //rof
  } //fi
  else if(x == 0){
    for(n=0;n<4;n++){
      if(ping())
          n--;
    } //rof
  } //fi
  else if(x == 3){
    for(n=0;n<6;n++){
      if(ping())
          n--;
    }
  }
  else{
    printf("This feature is not available yet!\n");
  }
  exit(0);
```

```
}
```

The header files are from the same book I used (Hacking: The Art of Exploitation). The program didn't change much from the last report besides bits here and there.

5. Conclusion

The project is nearly complete. All that is left is to attach the sensor properly to the coffee machine, build a box to hold all the circuitry and finish writing the application. If time permits, I'll write a basic GUI for the application and a simple install script so that you may run it as a bash command. Again, if time permits, I will try and see what was the problem with the Ethernet controller reception of packets.