EtherCoffee

...revolutionize your coffee

Project Manuel
Author: Calvin Ouellet-Ference
Spring 2010

Table of Contents

1.0 – Introduction

EtherCoffee is a 802.3 compliant device which connects to any network through CAT5 or CAT5e cable. EtherCoffee is operated from any system running Unix or any of it's open source derivatives. With the appropriate software, you can control EtherCoffee either through the command-line interface or the graphical user interface. It operates with a modified version of UDP datagrams on the registered port 9751 with the IP address of 192.168.0.225.

The concept for this device came with the notion that one day, everything will be given an IP address and be connected to the Internet. With this idea, EtherCoffee came to life with the purpose to satisfy anyone who do not want to interrupt their computing work to prepare coffee. The name EtherCoffee is derived from Ethernet which is the IEEE standard for computer networking. Thus making coffee through the Ethernet.

*
* *

2.0 – Setup and Operation

To setup EtherCoffee, please follow the following directions:

1. Place EtherCoffee in a safe area away from water or heat sources.

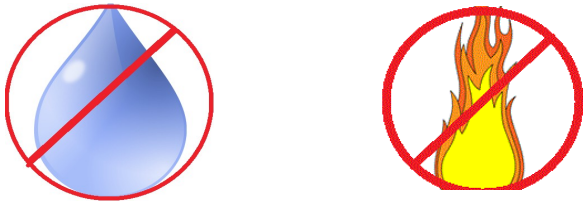

Figure 2.1 – Safety

2. Insert power cable in the appropriate plug located on the side

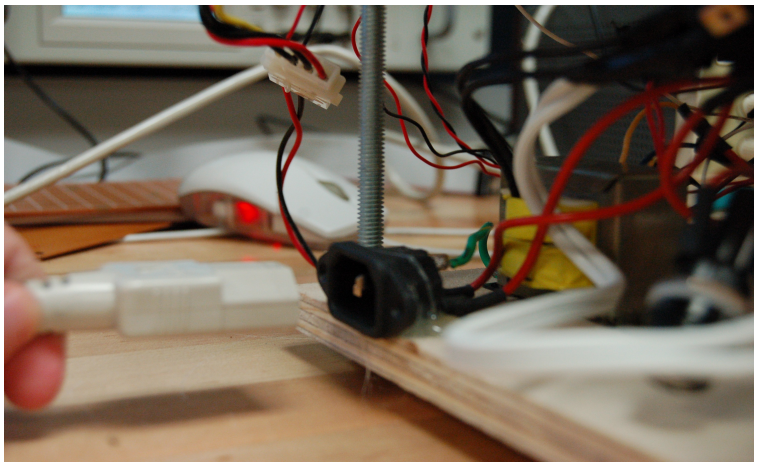

Figure 2.2 – Power Cable Connection

3. Insert Ethernet cable in the RJ45 jack on the side.

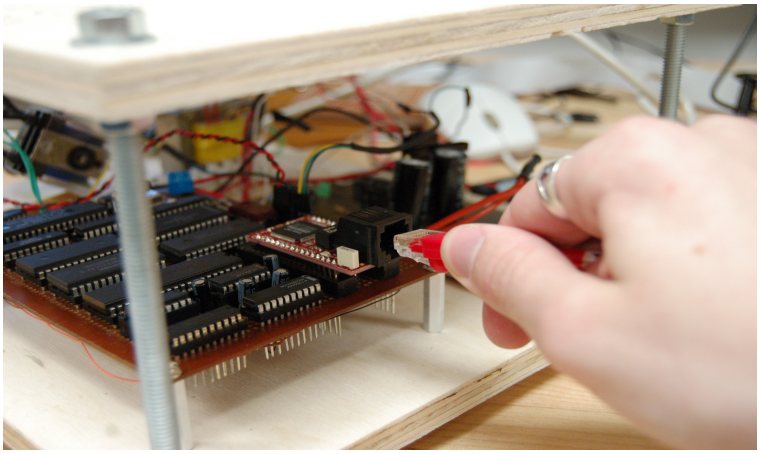

Figure 2.3 – Ethernet Cable Connection

4. Power-up EtherCoffee by pressing the power button located on the side.

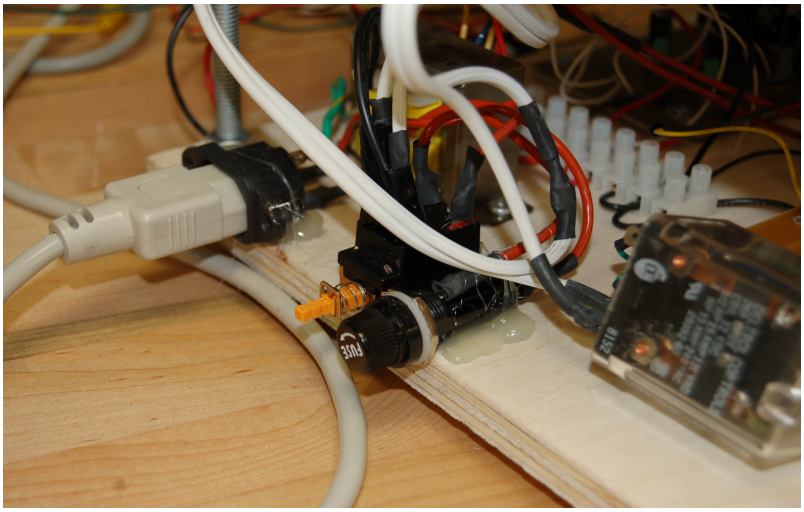

Figure 2.4 – Power Switch Location

5. On any Linux based operating system; run the install.sh script with superuser privileges by typing in the appropriate directory the following “sh ./install.sh”.

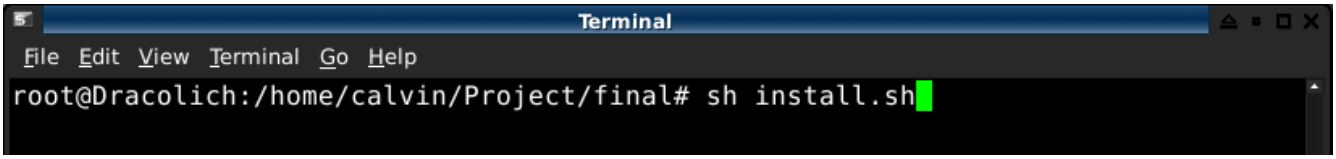

Figure 2.5 – Installation script execuation

6. Verify that EtherCoffee is responding properly by typing the following command: “eccheck”. If there is no reply, make sure step the previous steps were done correctly.

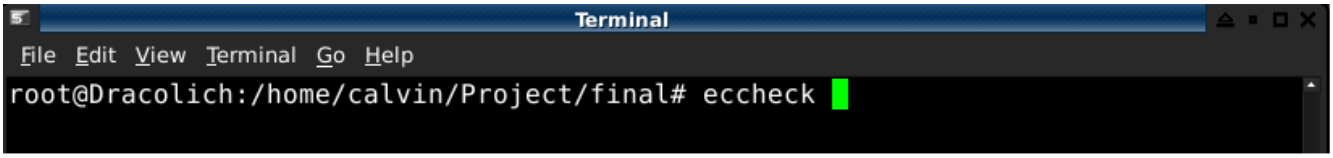

Figure 2.6 – eccheck command

7. Installation is complete, you may now begin brewing by typing “brew” at the command-line to be brought to the user interface.

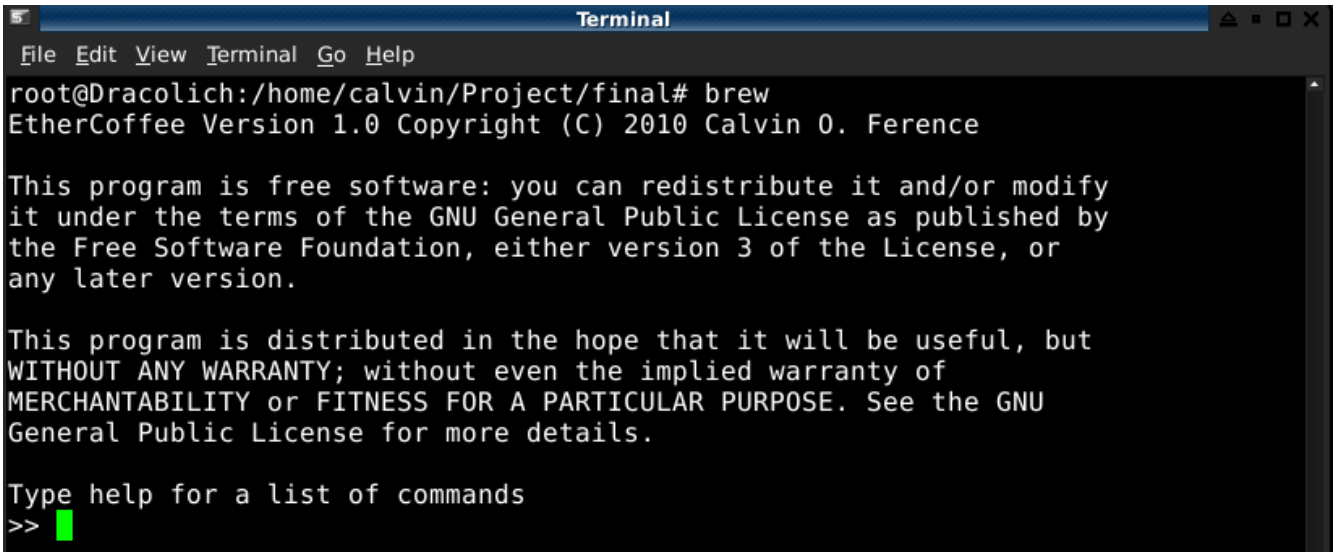

Figure 2.7 – CLI application

3.0 Specifications

AC to DC Power Supply

Input Voltage	16 VAC, 60 Hz
Output Voltage	+5 V, +12 V
Output Power	4.9 W
Standby Output Power	6.1 W
Total Power Dissipated	7.41 W

Intel 8088 Minimal System

Input Voltage	+5 V
Input Power	2.5 W

Coffee Machine

Water Tank Capacity	740ml
Input Current	5 A

General

Power requirements	120 VAC, 60 Hz
Power Consumption	615 W
Standby Power Consumption	14.94 W
Dimension	33cm x 31.5cm x 15.5cm
Weight	TBA
Supplied Accessories	None

4.0 Circuit Description

The nomenclature for describing memory locations is the following: 0xFFFF to express hexadecimal values.

EtherCoffee is built from four blocks:

- Application: Described in Section 5.2
- Ethernet Controller: Controls access to the network (Section 4.1)
- 8088 Minimal System: The heart of EtherCoffee, controls data processing and the various peripherals (Section 4.1)
- Coffee Machine: All the support circuitry for the Coffee Machine (Section 4.2 and Section 4.3)

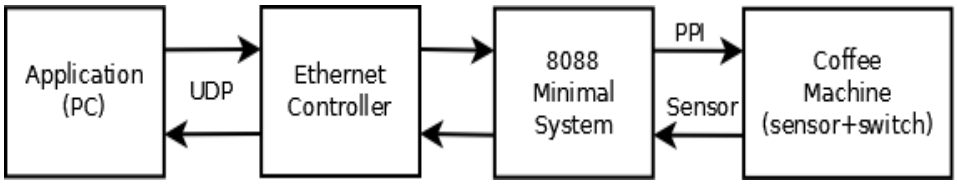

Figure 4.0.1 – Block Diagram of EtherCoffee

4.1 Intel 8088 Minimal System

The minimal system controls the entire system. It uses an Intel 8088 microprocessor clocked at 4.9MHz. The Intel 8088 is a 16-bit microprocessor with an 8-bit data bus and a 20-bit address bus and operates at positive 5 volts. Because the lower 8-bits of the address bus are multiplexed with the data bus, an 74HC373 latch is used with the microprocessors ALE signal to control when to latch the valid data. The firmware is programmed in the x86 Assembly language and the firmware itself is programmed in an Atmel 28C64B ROM. For temporary storage, the Hitachi 62256B SRAM is used though the firmware may be rewritten to not use the RAM but since it was already on the board, it was used. The memory map is partially decoded for RAM and ROM. The 4k RAM is located at the bottom of the memory map, between 0x0000 and 0x3FFF while the 8k ROM is located at the top of the memory map, from 0x8000 to 0xFFFF. The I/O Map is decoded in 8 sections of 200 bytes each. The I/O map ranges from 0x0000 to 0x0FFF. The entire decoding is done through a GAL G20V8B and the I/O is also done from the 74HC138 3 to 8 decoded. This decoder is what controls the chip enable signals on the various peripherals such as the PPI, the PIC and the Ethernet Controller.

For the minimal system to access a LAN, the CS8900A-H Ethernet Controller was used. Which is a pre-assembled board with all the relevant components included. It operates with a 20MHz clock and at positive 5 volts. It also has the transformer necessary to convert TTL signals into the Ethernet standard signals. It is bus compatible with the Intel 8088 and it can operate in both 8-bit and 16-bit mode though the 8-bit mode is used. It requires 4 address lines to be able to access the internal registers. Due to the nature of the pre-assembled board (See schematic in Appendix A) an 74HC245 buffer had to be added between the 8088 data bus and the CS8900A-H data bits. The upper 8 data lines were tied to ground to ensure that no noise corrupts data. Because DMA can not be used the DMA pins were left untouched. Nothing more was necessary to interface the CS8900A. The controller can be access through I/O address 0x0A00 which is Y6 of the 3 to 8 decoder.

There is an 8259 programmable interrupt controller (PIC) on the minimal system board. It has I/O address of 0x0400 and the chip select pin is attached to Y3 of the 3 to 8 decoder. It is used to simulate a real time clock. It has an 2400 Hz clock attached to IR0 from the 74HC4040. This is the only interrupt that is used because the CS8900A can not use interrupts in 8-bit mode.

The minimal system also has an 8255 programmable peripheral interface (PPI) which is used to control the relay with a TTL signal. It is the only application it is used for. It uses the bit 0 of PORTC to do so. It is located at I/O address 0x0200 and is activated by Y2 of the 3 to 8 decoder. There is also an 8251 UART and it's supporting logic on the minimal system, however it is currently not used. Finally, at memory location 0x0E000 there is a HEX display which is used for a quick visual aid of the status of EtherCoffee.

4.2 Relay

The relay is a generic power relay. It may take up to 15A of alternating current at 120VAC. It requires a +12V voltage to energize the coil. The coil is controlled by the minimal systems PPI. To get the PPI to control a +12V coil, a simple saturated 2N2222 transistor is used. In between the transistor and the PPI is a voltage follower. A single supply LM324 operational amplifier is used to accomplish this task. The base current of the transistor is of 4.167mA when activated. The relay is connected in series with the input power and act as an on/off switch.

4.3 Sensor

The purpose of the sensor is to detect the water level in the tank. The sensor used is the Sharp GP2D120. The GP2D120 is an IR sensor and it does continuous distance readings and returns the appropriate analog voltage depending on those readings. It's operating distance is of 4cm to 30cm which is plenty for this application. It comes in a pre-assembled package with three pins; power, ground and signal output Its operating range is from 4.5V to 5.5V. The sensor is mounted on the top of the tank pointing down. A Styrofoam floater covered in aluminum is used as an indicator of where the water level is currently at. The analog signal is fed into an ADC0804 analog to digital converter (ADC). It is an 8-bit ADC which is bus compatible with the Intel 8088. It uses 5V for the positive reference and ground for the negative reference. Two 1kΩ resistors are used as a voltage divisor for the Vref/2 reference pin. An 614.5 kHz clock was attached to the clock in pin of the ADC as the sampling rate. The ADC is located at I/O address 0x0800 which is activated by Y5 of the 3 to 8 decoder. The microprocessor can then send a write request to the ADC to begin conversion, and by the time the read request comes in, the ADC finished the conversion which takes approximately 1.62μs to complete. For better readings, a fan was added over the water tank to suck the steam out.

4.4 Power Supply Unit

The power requirements for EtherCoffee are positive 5 volts and positive 12 volts. The requirements are achieved with an 7805 and an 7812 voltage regulator. The 7805 is been driven to the limit with the current requirements of the minimal system of about 500mA. The power dissipated across the 7805 is approximately 5.82 W which requires a big heat sink. The power dissipated across the 7812 is approximately 1.577 W which requires a smaller heat sink. A fan was added to ensure a proper operating temperature. The power supply was designed to have a 25% ripple voltage but a much higher capacitor was added to ensure a proper ripple. Finally, some LED's were added at the outputs of the voltage regulators so that the user may see if the system is on or not. The design calculations for the power supply go as follows:

Specifications:
Supply Current: 120V AC at 60Hz
Output Voltage: +5V / +12V DC regulated
Output Current: 1.5A max
Output ripple: 25%

LM7805 and LM7812 were used.

Let:

V_I be the Input voltage
$V_{r(p)}$ be the ripple voltage
PIV peak value of the transformers secondary voltage
V_m max secondary voltage
V_D diode voltage drop

$V_I > (12\text{V} + 2\text{V}) = 14\text{V}$
$ripple = V_{r(p)} = 0.25 * 14 = 3.5\text{V}$
$V_{r(p-p)} = 3.5\text{V} \times 2 = 7\text{V}$
$V_{r(p-p)} = \frac{I_{DC}}{2fC} \rightarrow C = \frac{I_{DC}}{2fV_{r(p-p)}}$
$C = \frac{1.5\text{A}}{2 \times 60\text{Hz} \times 7\text{V}} = 1.785\text{mF}$
$2\ X\ 2200\mu F$ *was used*

$I_{DIODE} = 1.251 \cdot I_{DC}$
$I_{DIODE} = 1.8765\ A$

$PIV = V_m$
$V_m = V_{DC} + V_{7812\,DROP} + V_{r(p-p)} + 2\text{V}_D$
$V_m = 12\text{V} + 4\text{V} + 7\text{V} + 1.4\text{V} = 24.4\text{V}$

thus 1N4002 was used as it can support up to 2A of current.

5.0 Software Description

The source code for each module is found in Appendix B. Every program found in this manual is under the GNU General Public License.

The nomenclature used for describing data values is the following:

Binary : 0101b
Decimal : 5
Hexadecimal : 0x05

5.1 EtherCoffee Firmware

The firmware begins by initializing the various peripherals on the micro-controller as shown in Figure 5.1.1.

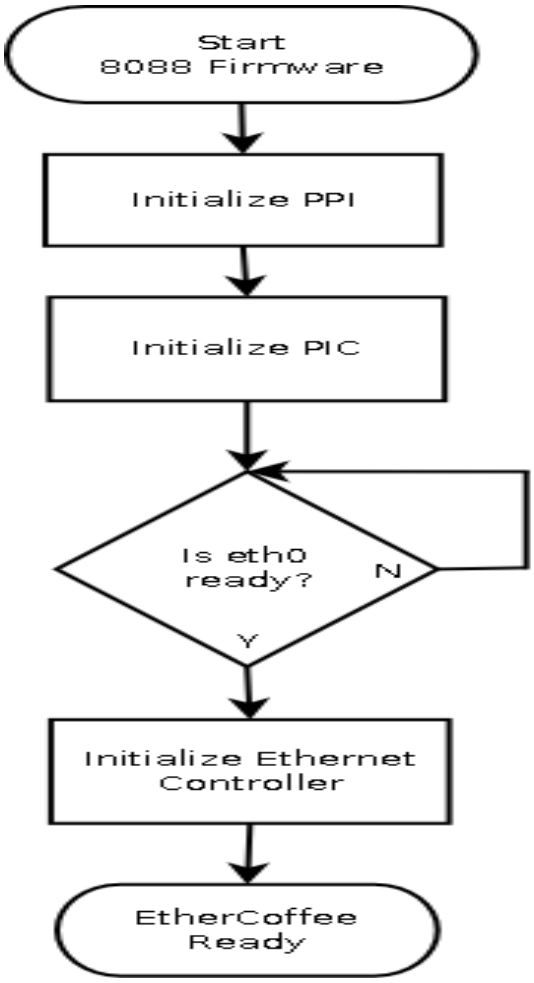

Figure 5.1.1 – Flowchart of Initializing Sequence

The firmware begins by initializing the 8255 programmable peripheral interface. It sets PORTA, PORTB and PORTC as output ports in Mode 0. This peripheral is used to control the relay from PORTC.

The firmware then initializes the 8259A programmable interrupt controller. It moves the rtc routine (Figure 5.1.2) into the interrupt vector 0x100. The Initialization Command Words used are ICW1, ICW2 and ICW4. The ICW1 is programmed in simple mode with the appropriate interrupt vector address and makes the PIC interrupt as edge trigger. ICW2 is programmed with vector number 0x40 and ICW4 is programmed to operate in 8086/8088 mode.

Finally the firmware waits for the Ethernet Controller to have finished it's internal calibrations before beginning to program it. It begins by programming the MAC address in the Ethernet Controller with the address of: 00:6F:66:66:65:65. It disables the Test Control Register and then it programs the Line Control Register (0x112) to tell the CS8900A to enable transmission and reception of packets. The EtherCoffee initialization is then complete.

*

* *

Figure 5.1.2 shows the real time clock interrupt routine.

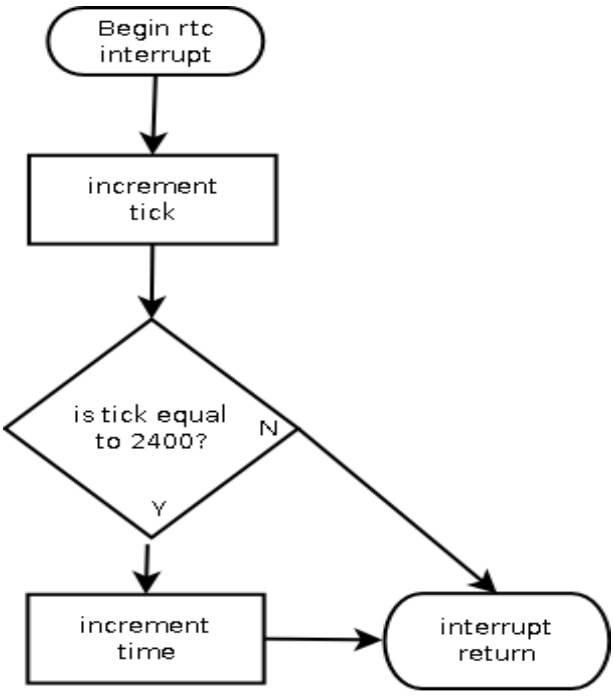

Figure 5.1.2 – Flowchart of the RTC routine

As mentioned, the PIC has an 2400Hz frequency attached to one of it's pins and each time it triggers, it counts one tick. Once there are 2400 ticks counted, it increments the time variable by one to show that one second has passed. This routine is quite important for the timeouts of the receive calls.

*

* *

Figure 5.1.3 shows the “main” of the firmware.

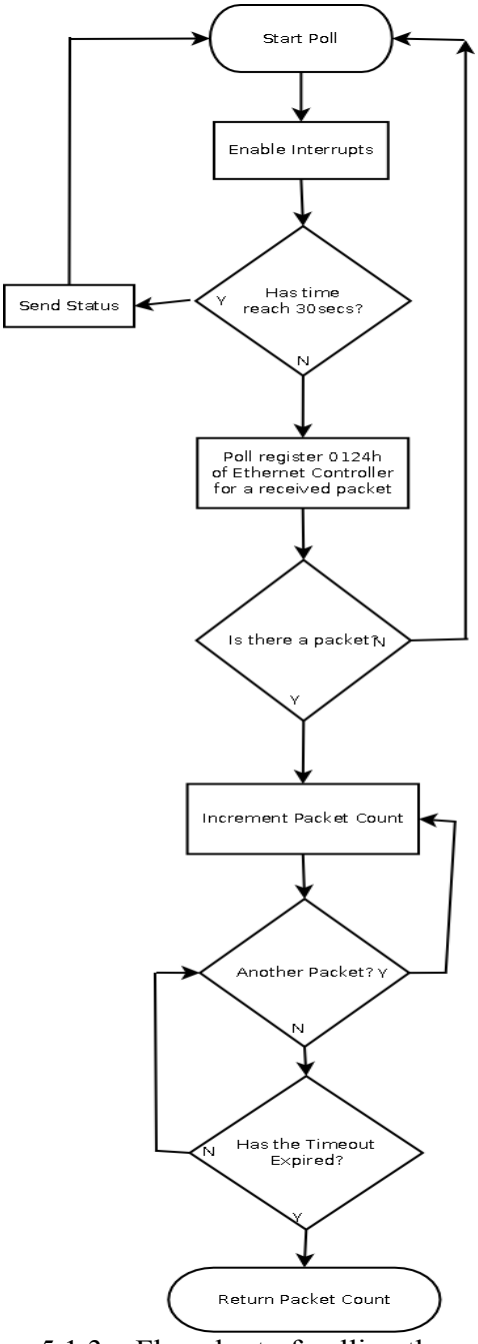

Figure 5.1.3 – Flowchart of polling the network

This section of the firmware begins by enabling interrupts. The purpose of this is to count thirty seconds, which after thirty seconds it sends out on the network an “alive” status. However, this was for the intent for a multi-threaded client which was not implemented on version 1.0 of the EtherCoffee software. It will then poll the Ethernet Controller and verify if a packet which passed the filters has arrived. If one was detected, it will reset the time variable and begin a five second timeout in which it will begin counting packet as it comes along. Depending on how many packet it has received, it will do certain functions.

This may seem an odd way for the communication process and it is. Due to time constraints and the lack of a TCP/IP stack in x86 assembly, the CS8900A had to be programmed with a basic one. However, when the CS8900A received a packet, it would not get passed into the receive buffer. Thus, when trying to read the receive buffer, only zero's and some stray data would appear.

*

* *

-

5.2 EtherCoffee Software

Figure 5.2.1 shows the main function of the EtherCoffee software.

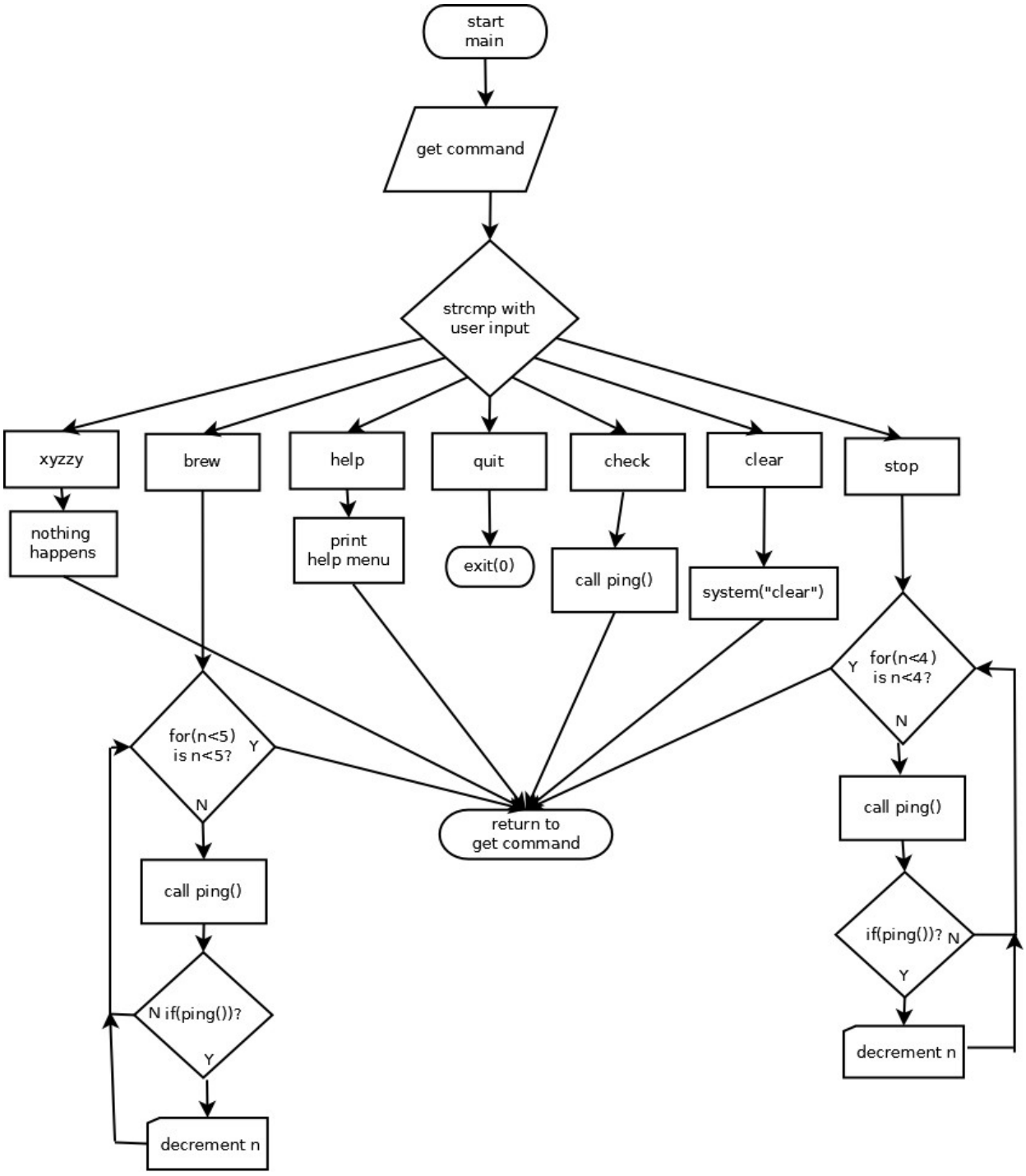

Figure 5.1.1 – flowchart of main()

The main is a simple function which displays a command-line like prompt so that user may enter different commands. Once the command finished executing, it will return to the beginning of the user input until the command quit or exit is entered. If the user enters the help command, it will display to the user the various commands he can use. The brew and stop commands will execute a for loop with their respective values. For example, the brew command will loop 5 times and it will execute the ping() function (see figure 5.2.3 for additional information). If the ping command is unsuccessful, it will decrement the loop value so it may execute it one more time. The reason for this is due to the protocol. Which will be explain in Figure 5.2.3. The clear command calls the command-line clear program to clear the terminal window of any clutter. Finally, the check commands sens a unique ping to EtherCoffee to verify if it is online on the network or not.

From here, the user can control EtherCoffee through the CLI. The GUI will be shown in section 5.3

*
* *

The send_udp function takes care of sending the messages to EhterCoffee. It opens a socket on port UDP 9157. If the open is unsuccessful, it will exit the program to prevent any complications. Otherwise, it will send the message “Brew me one!” over the network and EtherCoffee will interpret this appropriately. Once it is done, it closes the socket and returns to the application which called send_udp. Figure 5.2.2 will shows the send_udp() function.

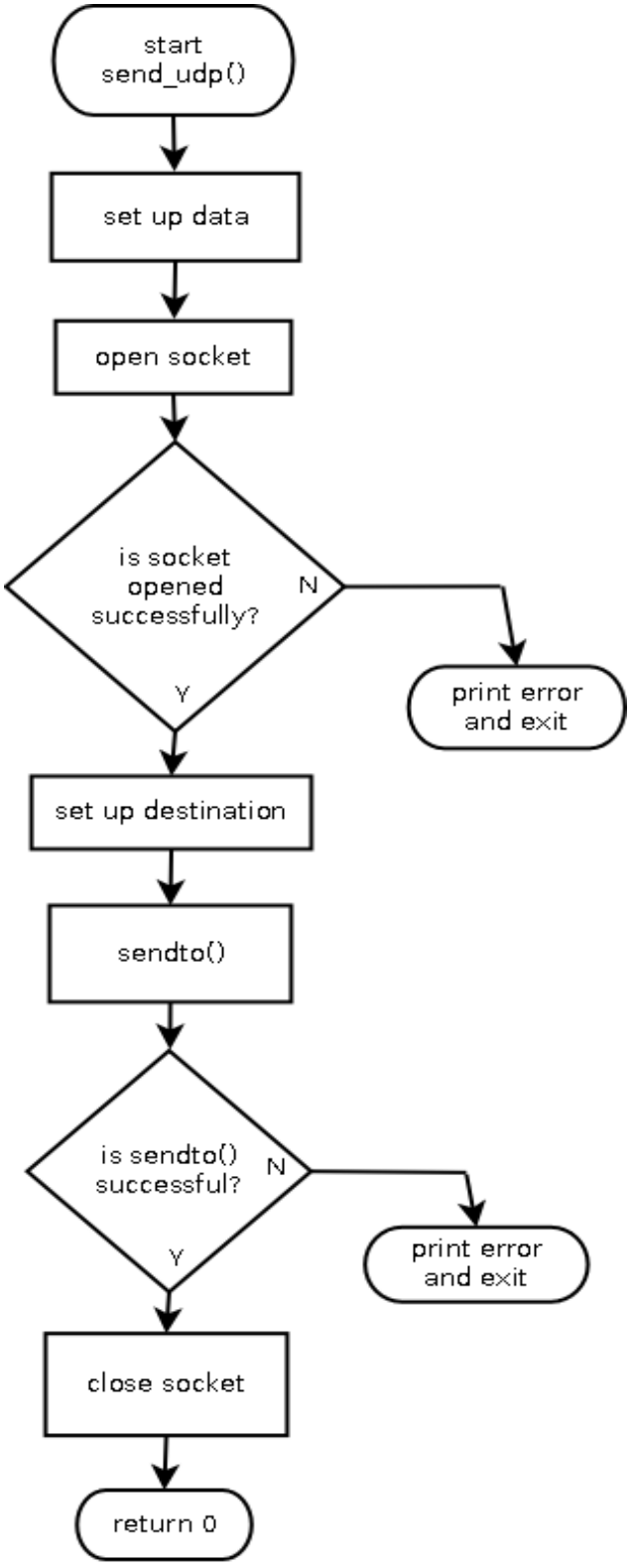

Figure 5.2.2 – flowchart of the send_udp function

Figure 5.2.3 shows the ping() function.

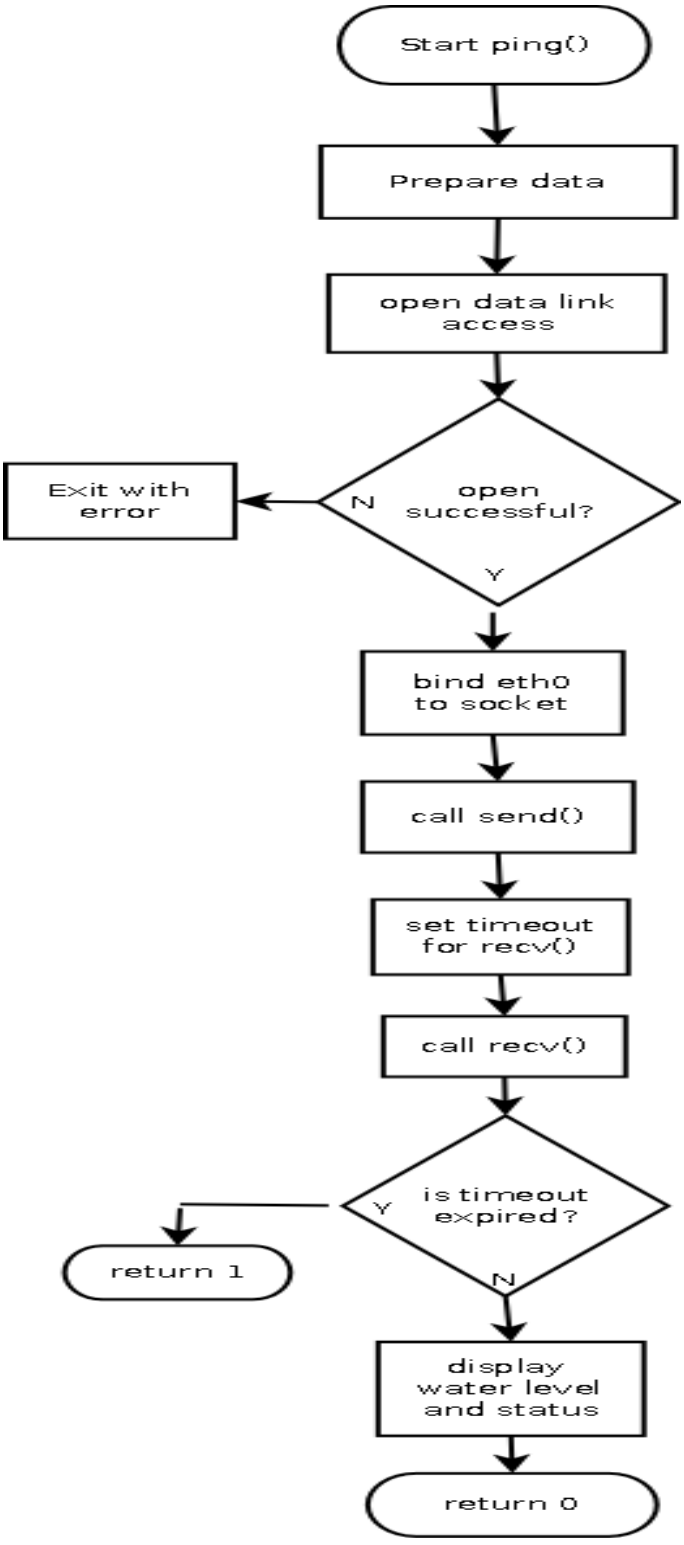

Figure 5.2.3 – flowchart of the ping function

The ping() function prepares the data link access to the network, calls the send_udp() function and awaits reply from EhterCoffee.

Due to various issues that arose while programming EtherCoffee's Ethernet controller, when the recv() call from the test program would pick up the packets sent out by EtherCoffee, the linux kernel would pre-process these packets before sending them to the recv() call. But the kernel would see these packets as “malformed” and would strip out what it think were the UDP, IP and MAC headers. To get around this issue, the program needs to get the packets before the kernel processes them, so a data link access needs to be created. This is the technique that various network sniffers use to capture data through the network. There is a library which could have been used to do this instead of manually doing it (the pcap library). However, the pcap library does not include timeouts in their recv() calls and no documentation has shown that you could use them. Which is the reason that the technique used currently is used. The only disadvantaged for this technique is the program needs superuser access. Thus needing to either “su -” or “sudo” at the CLI to use the application, or to log in as root to use the GUI.

Once the data link access has been created. The ping() function will bind the eth0 interface to the data link “socket” so that nothing from other network devices gets received. Finally, the program calls the send() function and awaits reply from EtherCoffee. If no reply has been received in a time of five seconds, the function will return a value of one to the function which called it. Otherwise, if something was received, it will display the water level and status to the screen and return 0.

*
* *

5.3 EtherCoffee Graphical User Interface Suite

Figure 5.3.1 shows the layout of the GUI.

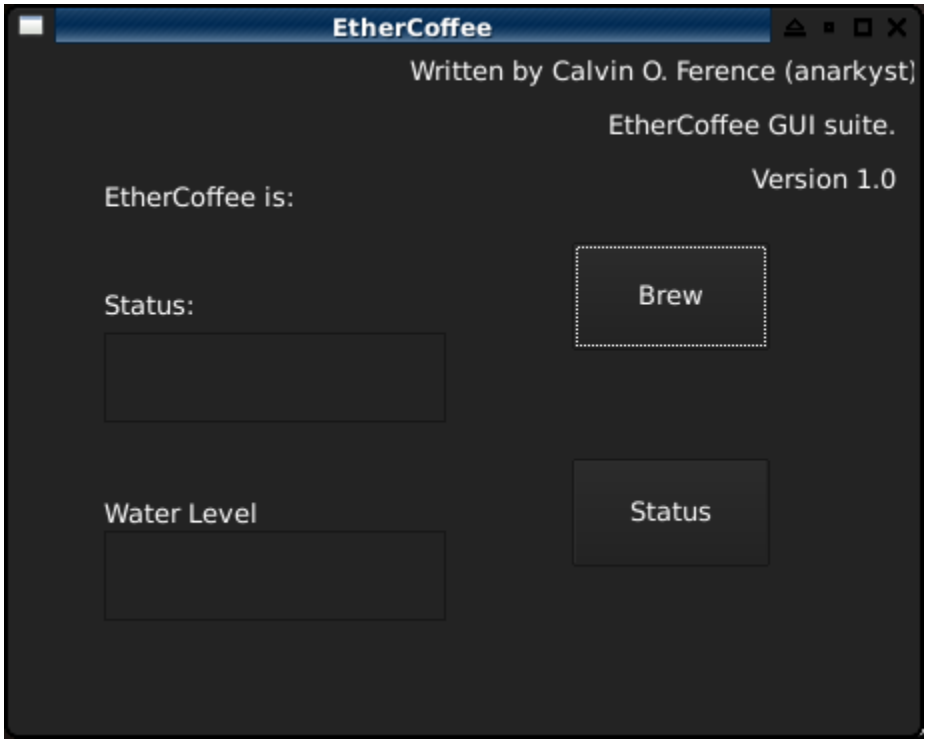

Figure 5.3.1 – GUI Layout

Figure 5.3.3 will show how the "status" button works and figure 5.3.4 will show how the "brew" button works. The GUI is written under the Gambas language (Gambas Almost Means BASic). It is a Visual Basic like language which is even driven. However compared to VB, it has no direct interfacing for applications written in C. To resolve this issue, the GUI was kludged together for it to work correctly. It uses modified functions of section 5.2 called ecbrew and eccheck (See Appendix B for source) which by themselves can be used from the command-line.

*
* *

Figure 5.3.2 shows the flowchart for the Form_Open class. The hole purpose behind this process is to verify that status of EtherCoffee. This uses the eccheck program mentioned above to verify if it is "Online" or "Offline". This gives the user a good idea from the start of the GUI if EtherCoffee is operational or not; assuming the user turned on EtherCoffee before using the software.

:

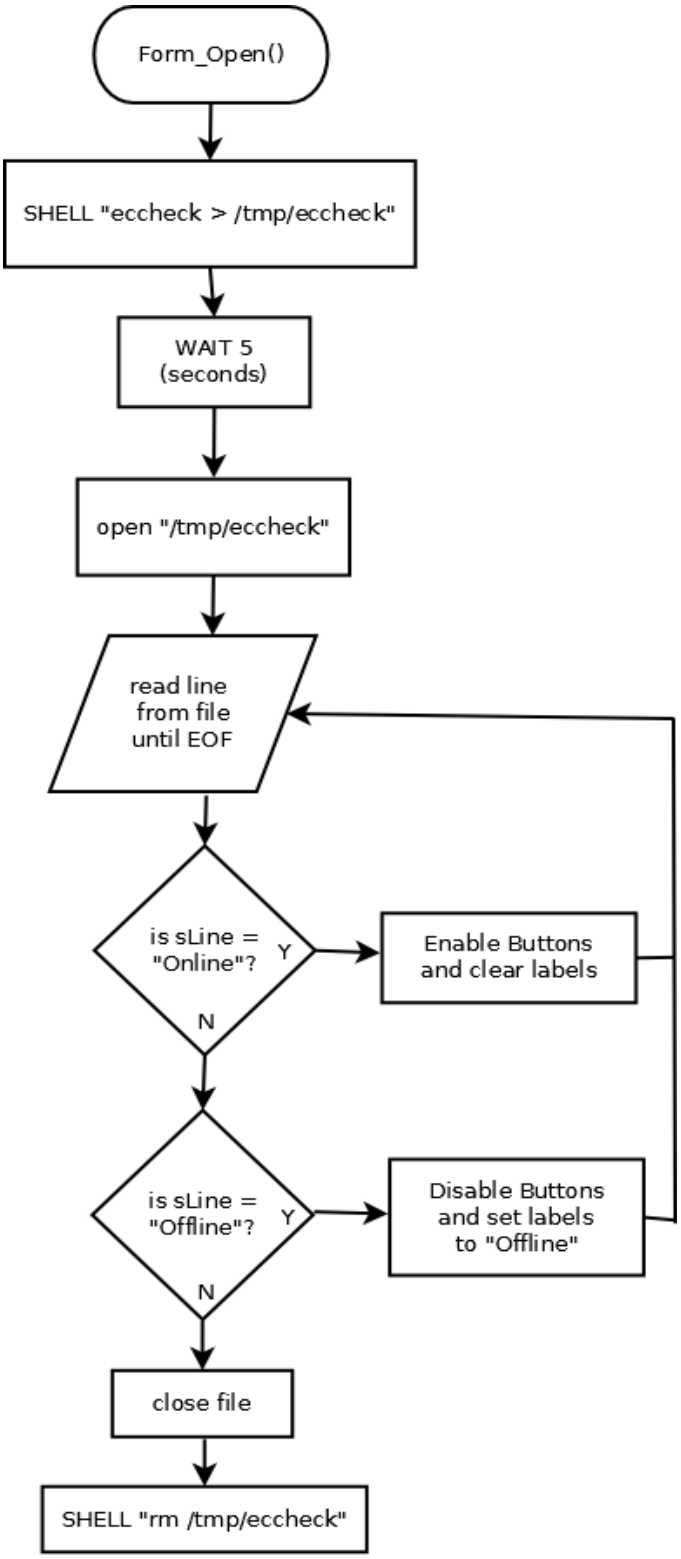

Figure 5.3.2 – Form_Open() flowchart

Figure 5.3.3 shows the “status” button subroutine:

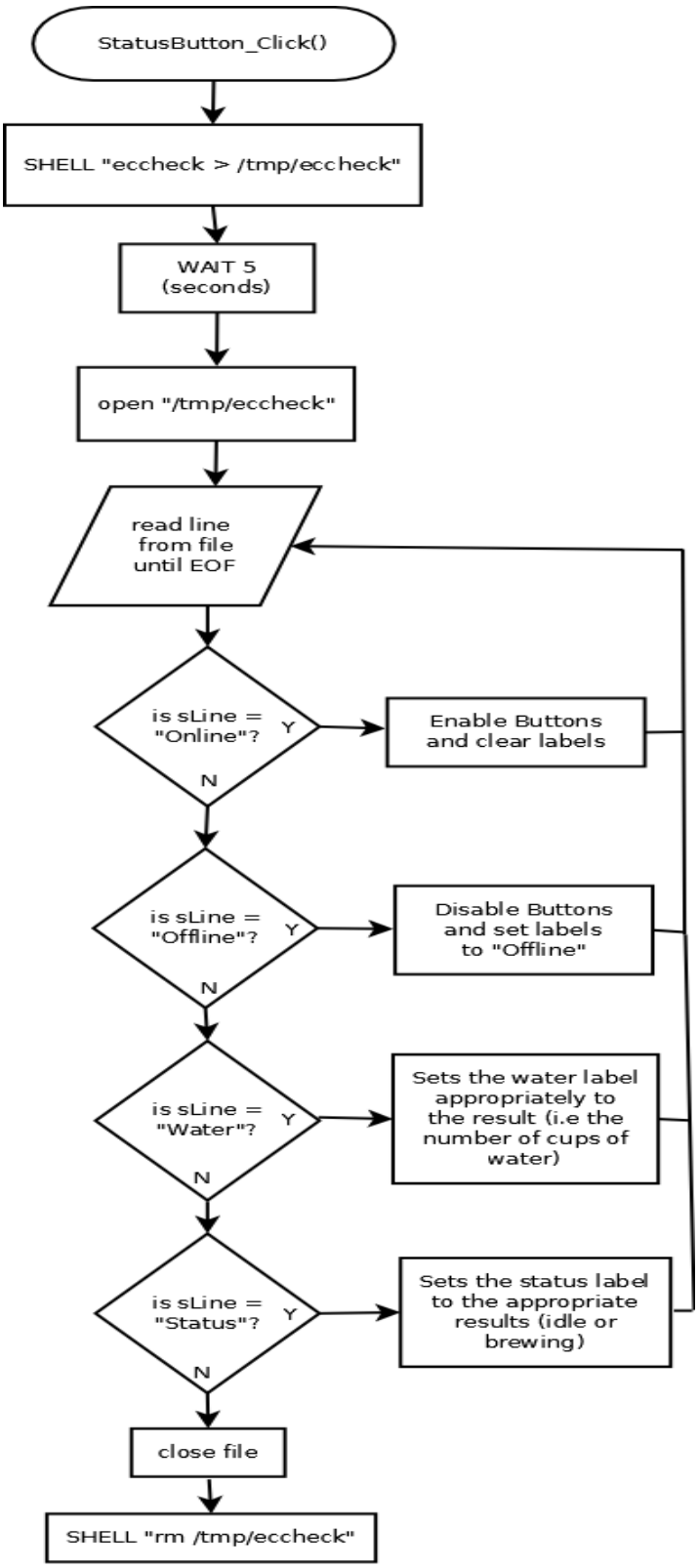

Figure 5.3.3 – Status Button Flowchart

This subroutine is almost the same as the Form_Open() routine. The only difference is it also verifies the water level and status. However, this button is essential if EtherCoffee did not respond to the original query. If the program does not detect that EtherCoffee is online, it will not enable the “Brew” button thus not being able to make any coffee. Figure 5.3.4 shows the “Brew” button routine, which is not much.

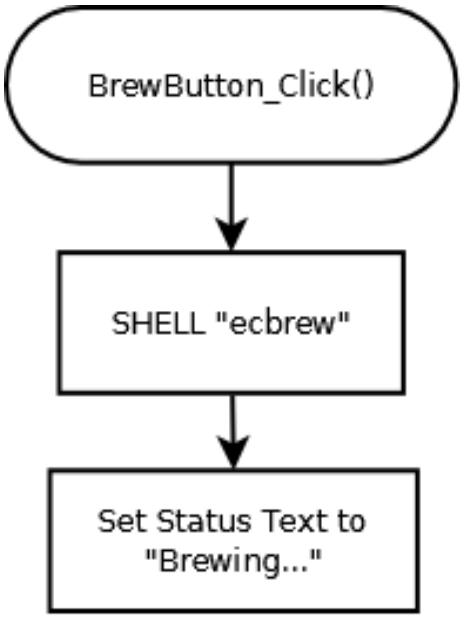

Figure 5.3.4 – Brew Button Flowchart

The program assumes that if the Brew button was enabled, then it is safe to transmit a brew command to EtherCoffee. However, if EtherCoffee goes offline, any attempt to “brew” will cause the program to hang as there is no statement to verify a socket timeout.

5.4 Installation Script

All the installation script does is compile the source code and move the executable files to the /usr/bin folder. The install creates three executables: brew, ecbrew and eccheck. The script requires bash to run (standard on most Linux distributions).

6.0 Calibration

Proper calibration of the analog components of EtherCoffee are essential for EtherCoffee to function properly. What must be calibrated is the floater located in the water tank so that the IR sensor may pick up the appropriate signal it sends out. The calibration is done by the following steps:

1. Make sure the "walls" for the floater are properly positioned as shown in the figure below (at about 125° angle).

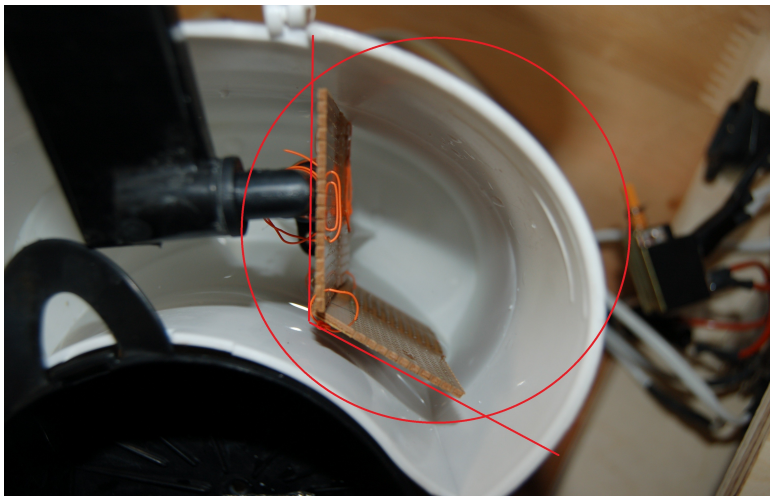

Figure 6.1 – "Wall" Position

2. Make sure the floater is located inside the "walls" as shown below:

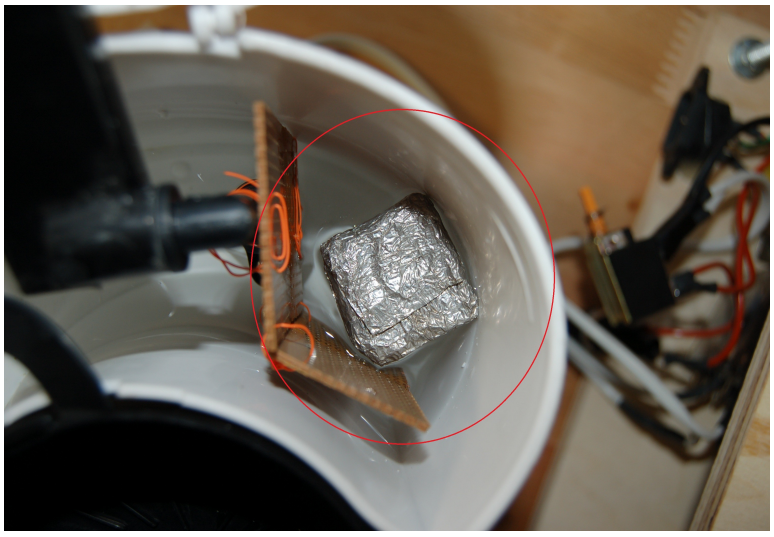

Figure 6.2 – Floater position

3. You may verify that it is properly calibrated by using the “water” command from the supplied software. Verify it when both the water tank is empty and full. If they turn out correct, then no further calibration is needed. Otherwise, repeat step 1 and step 2 and try again.

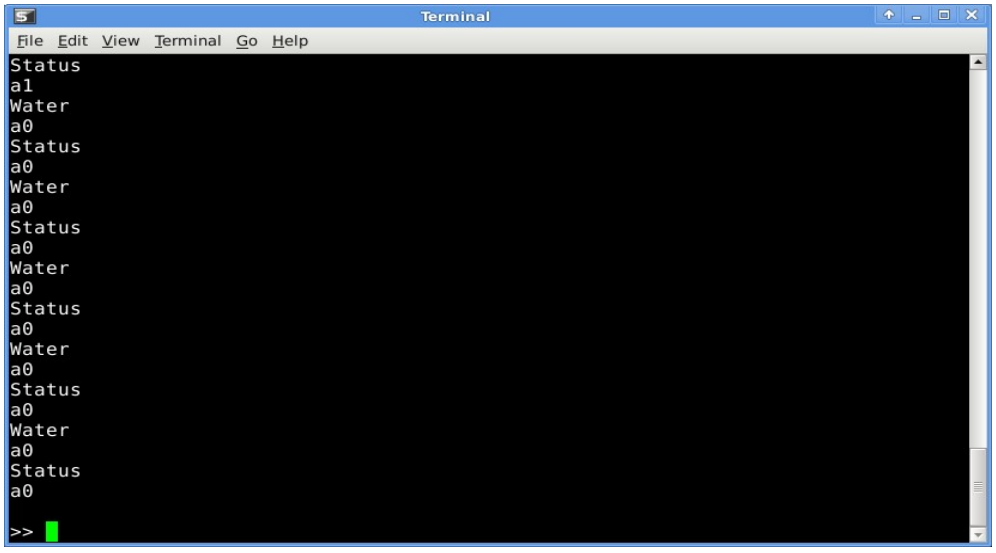

Figure 6.3 – CLI Water command

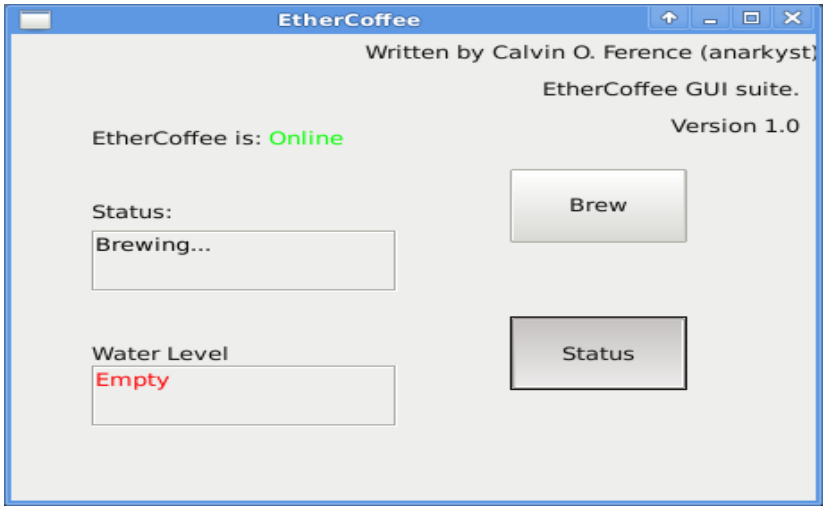

Figure 6.4 – GUI Water command

7.0 Troubleshooting

Problem: The device does not seem to be on.
Solution: Make sure the power cable is plug in properly and the power switch is switched on.

Problem: The power cable is correctly plug in but pressing the power switch does nothing.
Solution: Make sure the fuse is not blown. If it is blown, replace it with a spare with a minimum current rating of 10A at 250V.

Problem: EtherCoffee is telling that the water tank is empty even though it is not.
Solution: Make sure the IR sensor is properly connected (blue cable to the right). You may verify that the IR sensor is properly working by using a digital camera and looking at the lens. The lens should be a bright purple color as shown below.

Figure 7.1 – Verification of IR sensor

Problem: EtherCoffee is not giving the correct water level.
Solution: Verify that the water tank fan is functioning.

Problem: The device is on and the link LED is on (yellow), but no communication is happening.
Solution: Press the reset switch located on the minimal system (see Appendix A for illustration) and try again. A "2" will show on the display of the minimal system when it is in standby mode after a fresh boot up.

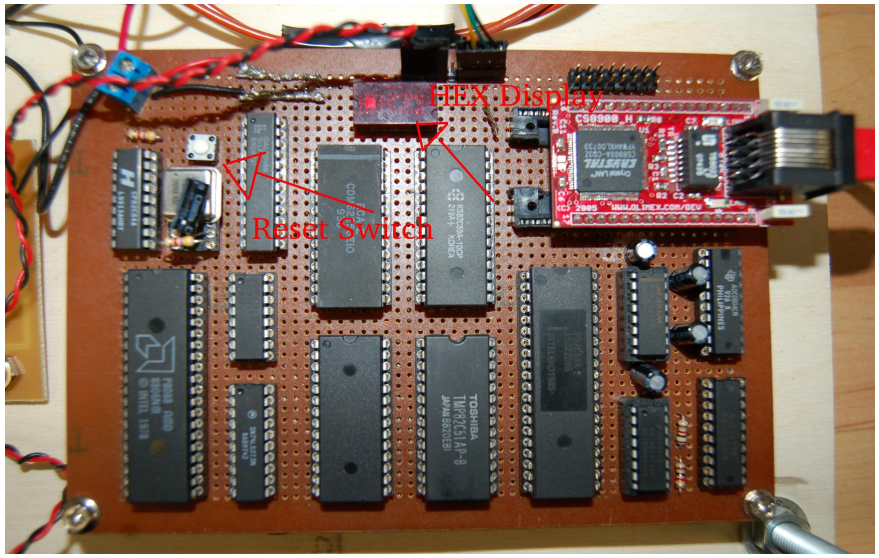

Figure 7.2 – Location of reset switch and hex display

Problem: While running the installation script; I get the following errors: "cp: cannot create regular file '/usr/bin/brew': Permission denied".
Solution: Make sure you have superuser privileges before installing or are in the sudoers.

Problem: While trying to use the "brew" command, I get the following errors:
"sudo: ifconfig: command not found"
"sh: arp: command not found"
Solution: Some Linux distrbutions (such as Slackware) will not permit you to use certain system commands even though you have sudo privileges. Try running the command as root.

Problem: The GUI does not seem to work.
Solution: For the GUI to function, you absolutely must have the roots X server running. In other terms, Ubuntu users will only be able to use the CLI, which any real Linux user use anyway.

Appendix A
Power Supply

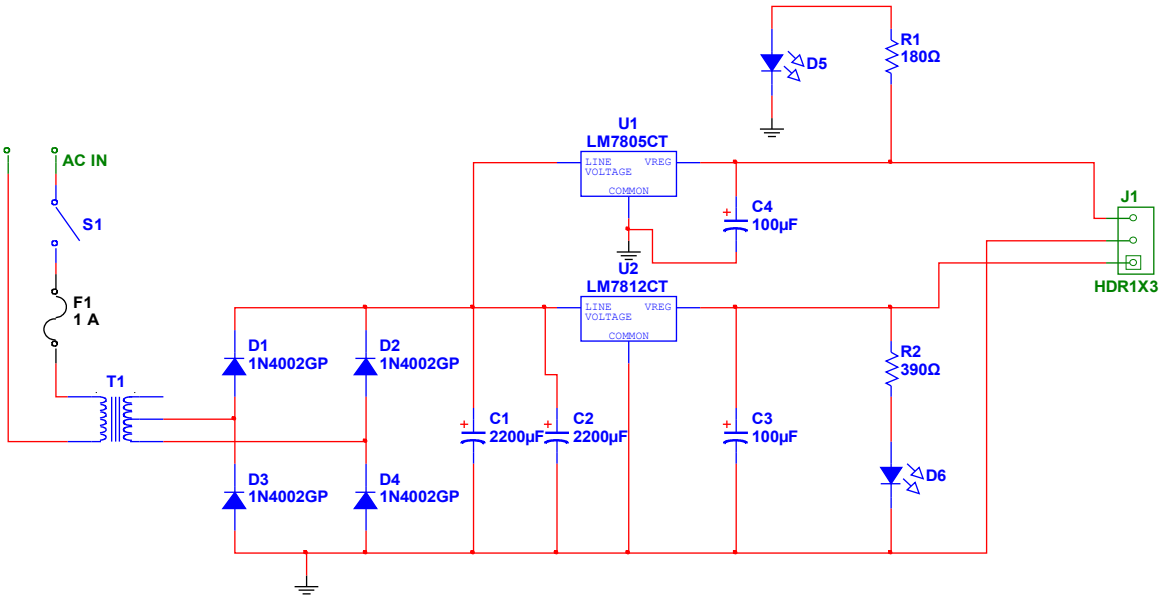

Figure 1 – Power Supply Schematic

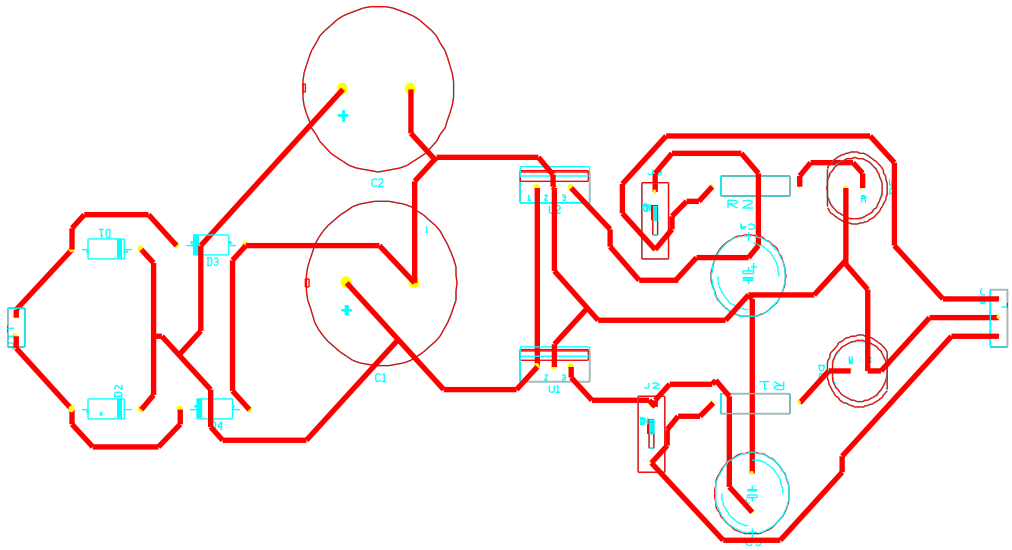

Figure 2 – Power Supply PCB Layout

Switch

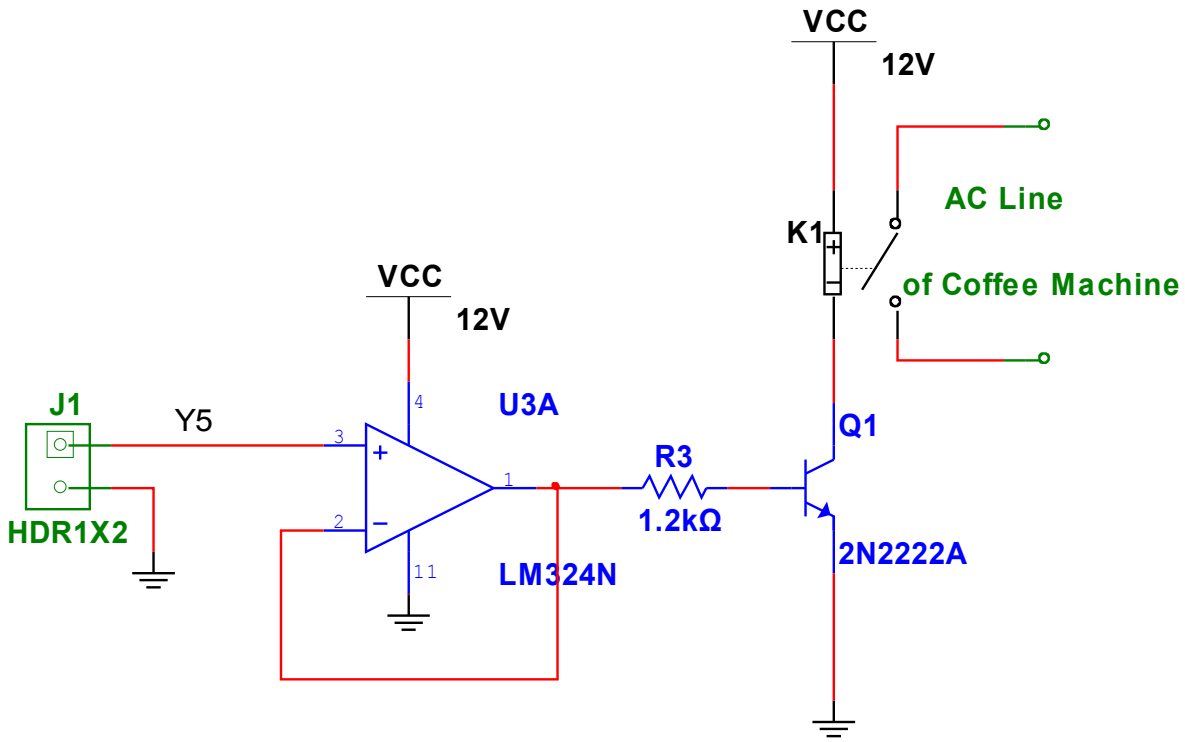

Figure 3 – Switching Circuitry Schematic

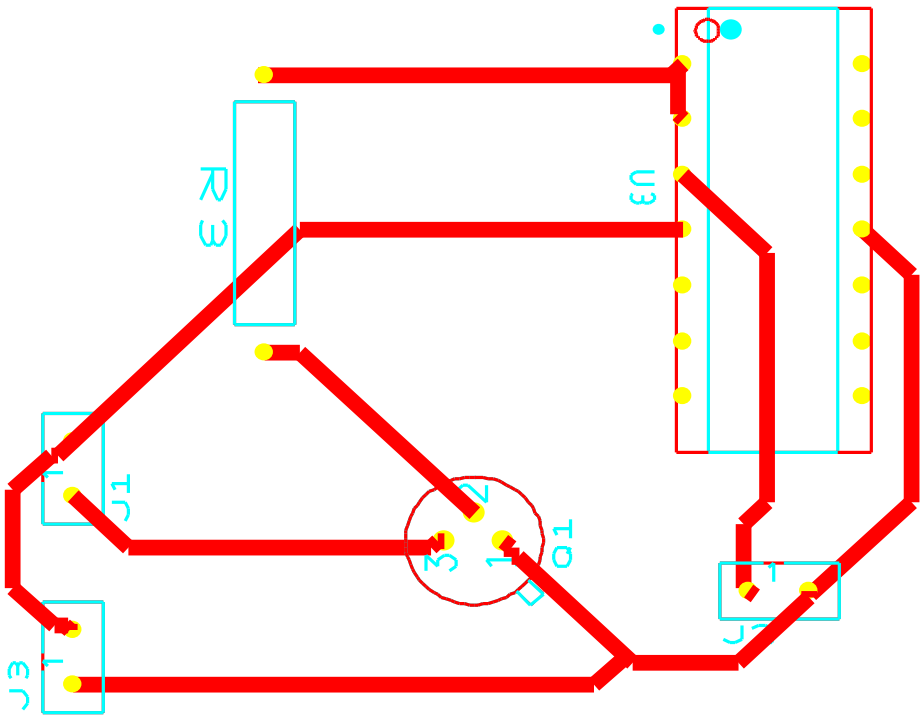

Figure 4 – Switching Circuitry PCB Layout

Intel 8088 Minimal System

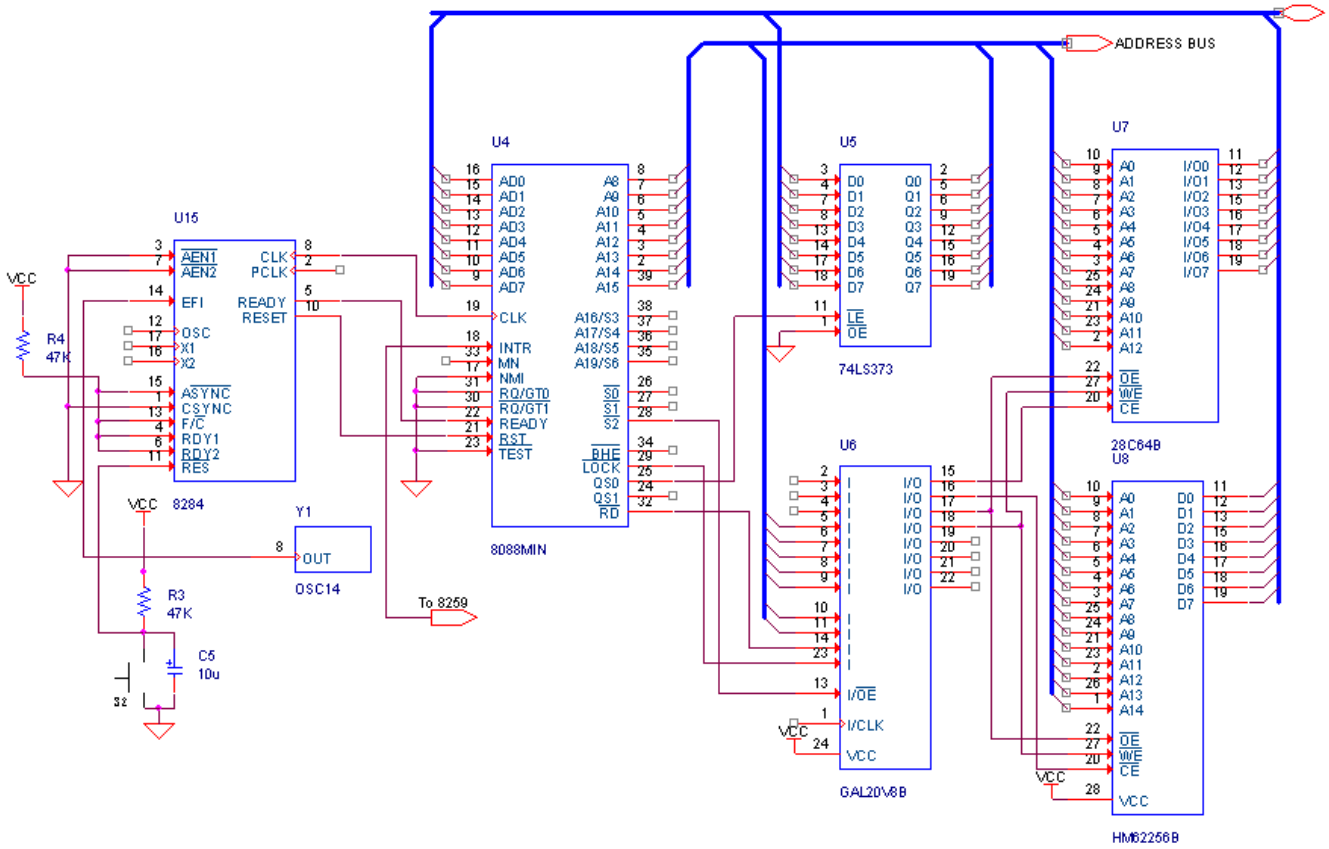

Figure 5 – Schematic of the Intel 8088 Microprocessor with memory and glue logic

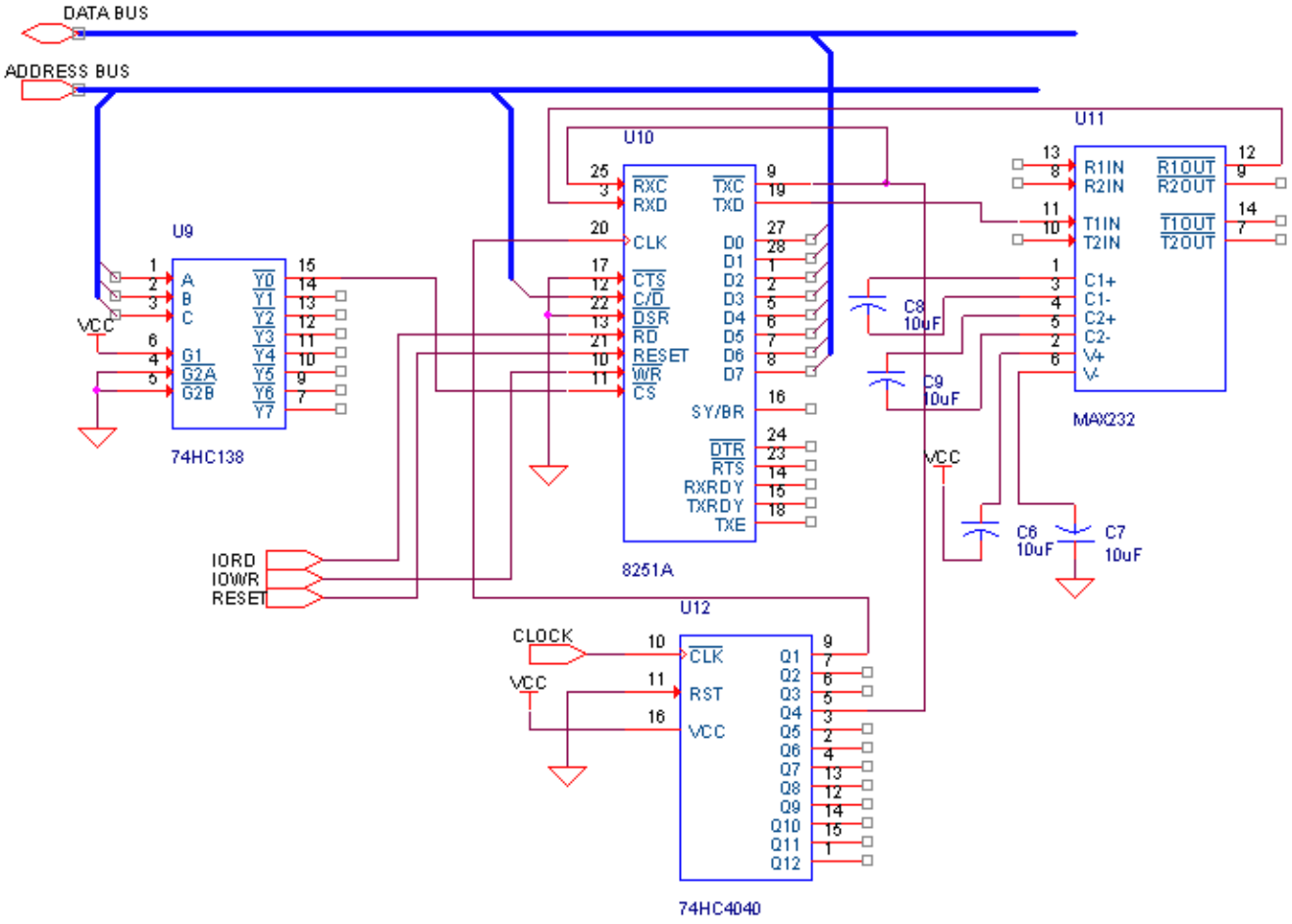

Figure 6 – Schematic of the UART

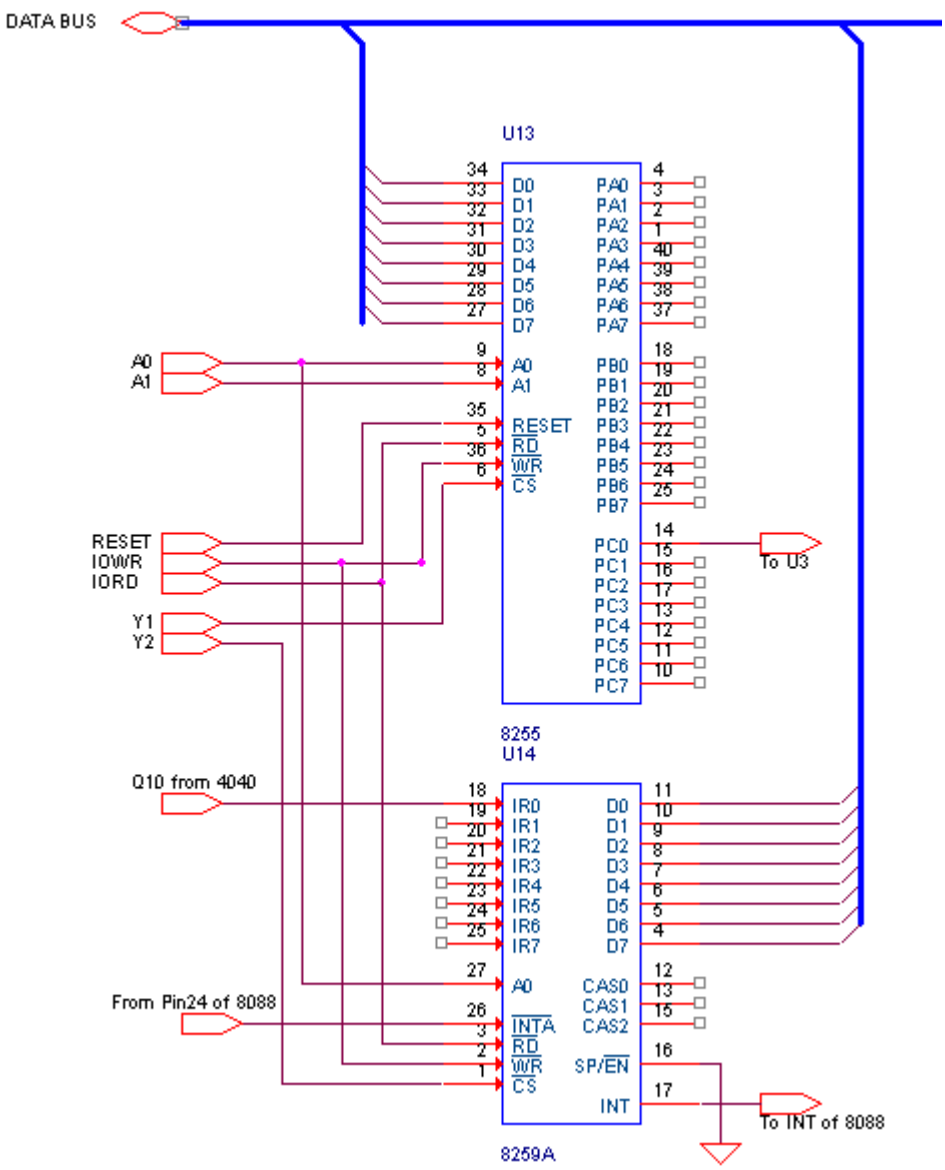

Figure 7 – Schematic of the PPI and PIC

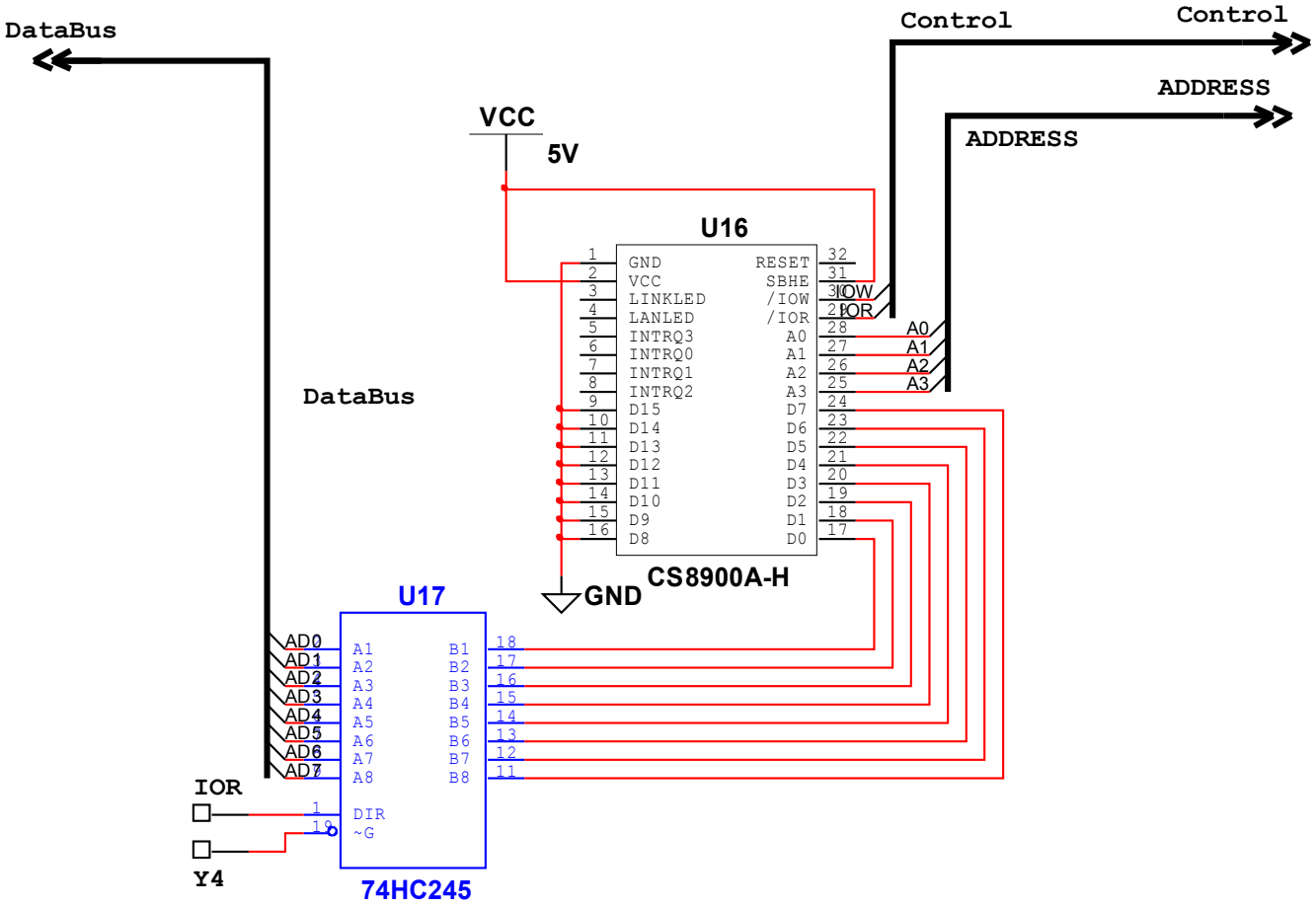

Figure 8 – Schematic of the Ethernet Controller

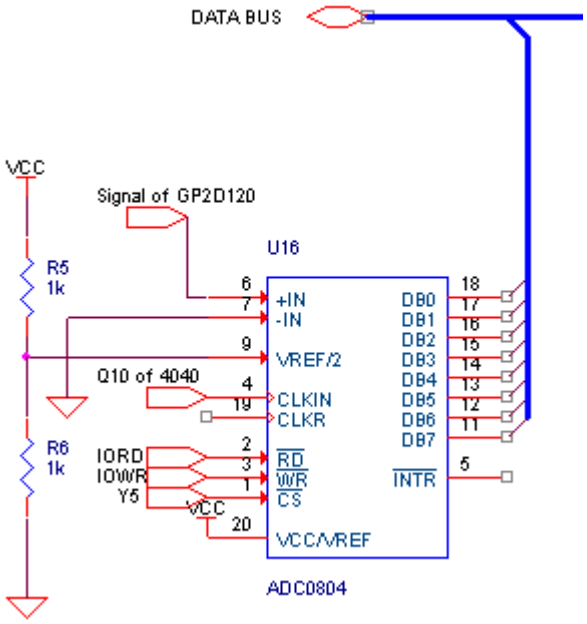

Figure 9 – Schematic of ADC

EtherCoffee

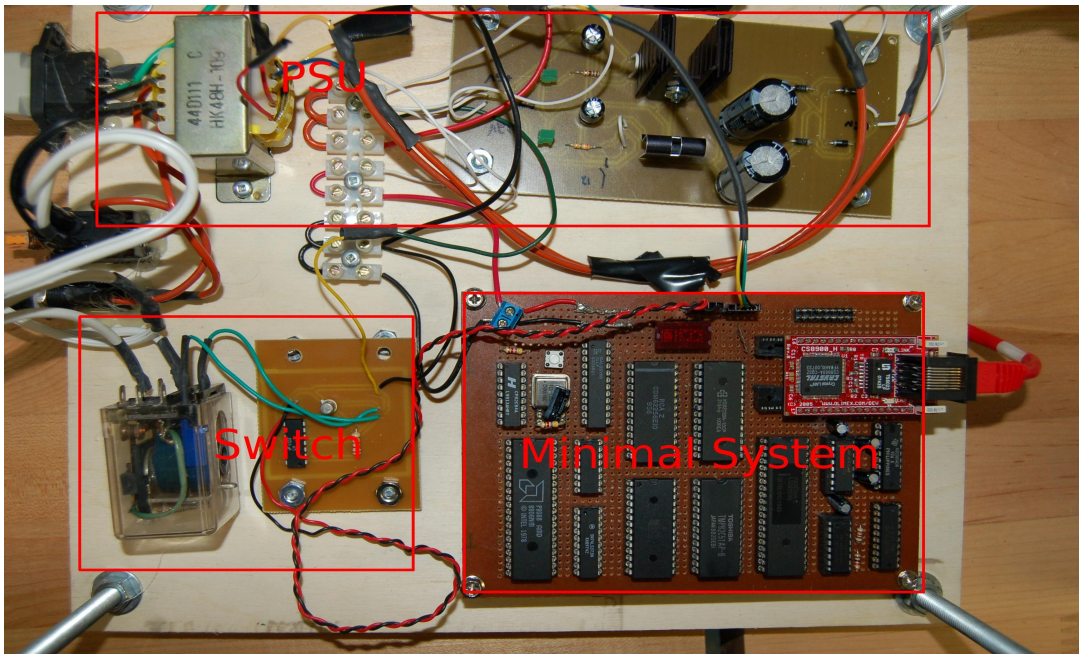

Figure 10 – Circuitry Layout

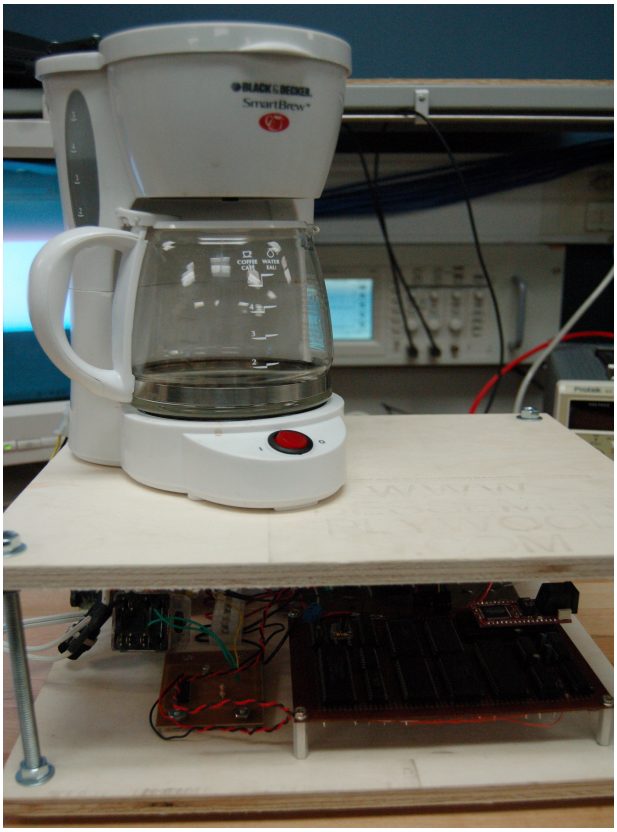

Figure 11 - EtherCoffee

Appendix B
EtherCoffee Firmware

```
; This program is free software: you can redistribute it and/or modify
; it under the terms of the GNU Public Liscence, or by
; the Free Software Foundation, either version 3 of the License, or
; any later version.
; This program is distributed in the hope that it will be useful, but
; WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY of FITNESS FOR A PARTICULAR PURPOSE. See the GNU
; General Public Liscence for more details.
;
; You should have received a copy of the GNU General Public Liscence
; along with this program. If not, see <http://www.gnu.org/licences/>.

; EtherCoffee Firmware v1.0 Copyright (C) 2010 Calvin O. Ference

.model tiny
.code

ROM     equ     0E000h
;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

; #defines
DSTATUS         equ         0A0h
DIDLE           equ         0A1h
DBREW           equ         0A2h

DWATER          equ         0B0h

DPROTO          equ         0C0h
DACK            equ         0C1h

      org     1000h
              ; eth0 Data
mac             db 00,00,00,00,00,00
data      db 00,00
chk_sum   db 00, 00

;RTC
tick      dw    0
time      db  0
```

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

; Water Level storage area... hmmm that sounds military for some odd reason
wlevel                  dw      0000h
water           db      00h

                org                     0E000h
main    proc
init:
                cli
                mov             ax,0
                mov             ds,ax
                mov             ss,ax
                mov             es,ax
                mov             sp,7ffh

                mov             di,1000h
                mov             al,0
zeroing:
                mov             [di],al
                inc             di
                cmp             di,10ffh
                jbe             zeroing
init_ppi:
                mov             ax,0
        mov     al,10000000b    ;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
poll:
        mov     al,02h
        mov     dx,LED
        out     dx,al

        sti
        cmp     time,30
        jae     send_status
        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
        call    recv
        cmp     p_cnt,6
```

```
                je          water_level
                cmp         p_cnt,5
                je          brew
                cmp         p_cnt,4
                je          stop
                jmp         poll
send_status:
                mov         al,0ffh
                mov         dx,ADCON
                out         dx,al
                mov         dx,ADCON
                in          al,dx
                cmp         al,42h
                jbe         emptzor
                cmp         al,52h
                jbe         twozor
                cmp         al,6Bh
                jbe         threezor
                mov         water,40h
                jmp         continuzor
threezor:
                mov         water,30h
                jmp         continuzor
twozor:
                mov         water,20h
                jmp         continuzor
emptzor:
                mov         water,00h
continuzor:
                cli
                mov         time,0
                mov         al,water
                mov         data,al
                cmp         brewing_flag,0
            je          status_set
                mov         al,DBREW
                mov         data+1,al
                jmp         status_go
status_set:
                mov         data+1,DIDLE
```

```
status_go:
                call    send
                jmp             poll

; Water Checking Sequence
; IS HERE!!
water_level:
                mov             cl,0
blarg:
                mov             dx,ADCON
                mov             al,0ffh
                out             dx,al
                mov             dx,ADCON
                in              al,dx
                mov             data+1,al
                inc             cl
                cmp             cl,9
                jbe             blarg
                jmp             poll

; The Brewing Sequence
; IST HERE MEIN KINDER!
brew:
                cli
                mov             time,0
                mov             dx,ADCON
                mov             al,0ffh
                out             dx,al
                mov             dx,ADCON
                in              al,dx
                cmp             al,42h
                jbe             nope
ppi_init_2:
                mov             ax,0
        mov     al,10000000b    ;configuration word for the 8255
                                ;both group A and B = mode 0
        mov     dx,PPICTL
        out     dx,al           ;send the configuration word
                mov             brewing_flag,1
rawr:
```

```
        mov     dx,LED
        mov     al,09h
        out     dx,al

        mov     time,0
        mov     dx,ADCON
        mov     al,0ffh
        out     dx,al           ;start ADC conversion

        push    cx
        mov     cx,100
x1:                             ; Many Thanks to
        dec     cx              ; Mr. Markou for this section
        jnz     x1
        pop     cx
        mov     dx,ADCON
        in      al,dx
        cmp     al,42h
        jbe     fini
        mov     dx,LED
        mov     al,0Bh
        out     dx,al
        jmp     rawr
fini:
        mov     dx,LED
        mov     al,0Ah
        out     dx,al
        sti
        cmp     time,60             ;55 secs
        jbe     fini
        cli
        mov     time,0
        mov     data+1,DIDLE
        mov     dx,PORTC
        mov     al,0ffh
        out     dx,al
        jmp     poll
nope:
        mov     data,0A0h
        mov     data+1,0A1h         ;need to arrange this
```

```
                call    send
                jmp             poll
; The Stopping Squence
; Ici a*** de t*******
stop:
                cli
                mov             al,0ffh
                mov             dx,PORTC
                out             dx,al
                mov             brewing_flag,0
                jmp             poll
rtc:
                inc             tick
                cmp             tick,FREQ
                jbe             idone
                mov             tick,0
                inc             time
idone:
                iret
main    endp

recv    proc
                sti
                mov             time,0
                mov             p_cnt,1
                call    send
recv_poll:
                cmp             time,5 ; 5 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:
                cli
                mov     al,p_cnt
                mov     time,0
                ret
recv    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
                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             ;AND HERE
;mov    al,data
out     dx,al
mov     dx,RXTX0H
mov     al,0a8h             ;HERE
;mov    al,data+1
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

    org   0FFF0h
reset db    0eah    ;jmp
    dw    ROM     ;ip
    dw    0000h   ;cs
    end
```

EtherCoffee CLI Suite
brew.h

```
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <net/if.h>
#include <linux/if_ether.h>
#include <arpa/inet.h>

#define DEBUG printf("[DEBUG]\n");
#define ADDR "192.168.0.225"
#define LOCAL_ADDR "192.168.0.176"
#define MAX 11
#define SA struct sockaddr
#define PORT "9751"

#define DSTATUS 0xA0
#define DIDLE     0xA1
#define DBREW   0xA2
#define DWATER 0xB0
#define DATA_BYTE_1 42
#define DATA_BYTE_2 43

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

int ping(void) {

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

```
   struct sockaddr_in myaddr;

   bzero(&myaddr, sizeof(myaddr));

   myaddr.sin_family = PF_PACKET;
   myaddr.sin_addr.s_addr = htonl(LOCAL_ADDR);
   myaddr.sin_port = htons(PORT);
   tv.tv_sec = 2;
   tv.tv_usec = 0;

   if((sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_IP))) == -1)
        fatal("in socket");
/*    if(setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv))){
        printf("setsockopt error\n");
        exit(1);}
*/      /*
   if((bind(sockfd, (SA*) &myaddr, sizeof(myaddr))) != 0){
     fatal("in bind");
     exit(1);
   }*/
   memset(&ifr, 0, sizeof(ifr));
   snprintf(ifr.ifr_name, sizeof(ifr.ifr_name), "eth0");
   send_udp();

   for(;;){
   if(setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv)) != 0){
        printf("setsockopt error\n");
        exit(1);}
   if(setsockopt(sockfd, SOL_SOCKET, SO_BINDTODEVICE, (void*)&ifr, sizeof(ifr))
!= 0){
        printf("setsockopt error\n");      // NEED TO BIND ETH0 TO THIS SOCKET!
        exit(1);}                          // IF WIFI IS ON, I SNIFF THE PACKETS FROM IT
   if((recv_length = recv(sockfd, buffer, 8000, 0)) <= 0){
         fprintf(stdout, "socket timeout\n");
        return 1;}
     if(buffer[0] == 00 && buffer[1] == 0x26 && buffer[2] == 0x2D){
        fprintf(stdout, "Water\n");
        fprintf(stdout, "%x\n",buffer[DATA_BYTE_1]);
        fprintf(stdout, "Status\n");
        fprintf(stdout, "%x\n",buffer[DATA_BYTE_2]);
```

```
        close(sockfd);
        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;
  struct timeval tv;

  tv.tv_sec = 2;
  tv.tv_usec = 0;

  if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0){
    fprintf(stderr, "Error while opening socket!\n");
    exit(1);}
  if(setsockopt(sockfd, SOL_SOCKET, SO_SNDTIMEO, &tv, sizeof(tv))){
    printf("setsockopt error\n");
    exit(1);} // hci control right here

  server.sin_family = AF_INET;
  inet_pton(AF_INET, ADDR, &server.sin_addr);
  server.sin_port = 9751;
  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){
    fprintf(stderr, "Error while sending message!\n");
    exit(1);}

  close(sockfd);
  return 0;}
```

brew.c

```
#include "brew.h"

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

  if(argc != 1){
    fprintf(stderr, "Too many arguments, aborting\n");
    exit(1);
  }
  int n, x;
  char select[MAX];

  system("sudo ifconfig eth0 192.168.0.176");
  system("arp -s 192.168.0.225 00:6F:66:66:65:65");

  fprintf(stdout, "EtherCoffee Version 1.0 Copyright (C) 2010 Calvin O. Ference\n\n");
  fprintf(stdout, "This program is free software: you can redistribute it and/or modify\n");
  fprintf(stdout, "it under the terms of the GNU General Public License as published
by\n");
  fprintf(stdout, "the Free Software Foundation, either version 3 of the License, or\n");
  fprintf(stdout, "any later version.\n\n");
  fprintf(stdout, "This program is distributed in the hope that it will be useful, but\n");
  fprintf(stdout, "WITHOUT ANY WARRANTY; without even the implied warranty
of\n");
  fprintf(stdout, "MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU\n");
  fprintf(stdout, "General Public License for more details.\n");
  fprintf(stdout, "\nType help for a list of commands\n>> ");
  for(;;){
    fgets(select, MAX, stdin);
    for(n=0;n<sizeof(select);n++){
      select[n] = tolower(select[n]);
    }
    if(strcmp(select, "brew\n") == 0){
      for(n=0;n<5;n++){
          if(ping())
            n--;
      }
      fprintf(stdout, "\n>> ");
```

```
    }
    else if(strcmp(select, "check\n") == 0){
      for(n=0;n<1;n++){
          if(ping())
            n--;
      }
    }
    else if(strcmp(select, "stop\n") == 0){
      for(n=0;n<4;n++){
          if(ping())
            n--;
      }
      fprintf(stdout, "\n>> ");
    }

    else if(strcmp(select, "water\n") == 0){
      for(n=0;n<6;n++){
          if(ping())
            n--;
      }
      fprintf(stdout, "\n>> ");
    }
    else if((strcmp(select, "quit\n") == 0) || ((strcmp(select, "exit\n") == 0))){
      fprintf(stdout, "Enjoy the coffee!\n");
      exit(0);
    }
    else if(strcmp(select, "clear\n") == 0){
      system("clear");
      fprintf(stdout, "Type help for a list of commands\n>> ");
    }
    else if(strcmp(select, "status\n") == 0){
      fprintf(stdout, "This feature is not yet available! Sorry! Mille Pardon! Gomen Nasai!
Verzeihen Sie! Sentímolo!\n>> ");
    }
    else if(strcmp(select, "help\n") == 0){
      fprintf(stdout, "\nEtherCoffee Version 0.3, Written by Calvin O. Ference
(w3b_wizzard)\n\n");
      fprintf(stdout, "(brew) - Tells EtherCoffee to make a tasty cup o' Jo\n");
      fprintf(stdout, "(check) - Checks if EtherCoffee is online\n");
      fprintf(stdout, "(clear) - Clears the screen\n");
```

```
      fprintf(stdout, "(help) - Prints this menu\n");
      fprintf(stdout, "(quit) - Exits this application\n");
      fprintf(stdout, "(status) - Gets EtheCoffees Status\n");
      fprintf(stdout, "(stop) - (Legacy) Overides the brew command\n");
      fprintf(stdout, "(water) - Checks the current water level of EtherCoffee\n>> ");
      }
    else if(strcmp(select, "xyzzy\n") == 0){
      fprintf(stdout, "nothing happens\n>> ");
    }
    else
      fprintf(stdout, "Invalid Command\n>> ");
  }
  exit(1); // If it got here... then there's a problem, thus FOOBAR!
}
```

EtherCoffee GUI Suite
FMain.form

```
# Gambas Form File 2.0

{ Form Form
  MoveScaled(0,0,50,38)
  Text = ("EtherCoffee")
  { Label1 Label
    MoveScaled(33,3,22,3)
    Text = ("EtherCoffee GUI suite.")
  }
  { Label2 Label
    MoveScaled(41,6,22,3)
    Text = ("Version 1.0")
  }
  { Label3 Label
    MoveScaled(22,0,31,3)
    Text = ("Written by Calvin O. Ference (anarkyst)")
  }
  { Label4 Label
    MoveScaled(5,13,6,3)
    Text = ("Status:")
  }
  { BrewButton ToggleButton
    MoveScaled(31,11,11,6)
    Text = ("Brew")
  }
  { StatusButton ToggleButton
    MoveScaled(31,23,11,6)
    Text = ("Status")
  }
  { StatusLabel TextLabel
    MoveScaled(5,16,19,5)
    Text = ("")
    Border = Border.Etched
  }
  { WaterLabel TextLabel
```

```
    MoveScaled(5,27,19,5)
    Text = ("")
    Border = Border.Etched
  }
  { Label5 Label
    MoveScaled(5,25,10,2)
    Text = ("Water Level")
  }
  { Label6 Label
    MoveScaled(5,7,11,3)
    Text = ("EtherCoffee is:")
  }
  { onofflabel Label
    MoveScaled(16,7,11,3)
    Text = ("")
  }
}
```

FMain.Class

```
' Gambas class file
PUBLIC online AS Boolean
PUBLIC water AS Integer

PUBLIC SUB Form_Open()
  DIM hFile AS File
  DIM sLine AS String

  SHELL "eccheck > /tmp/eccheck"
  WAIT 5
  hFile = OPEN "/tmp/eccheck" FOR READ
  WHILE NOT Eof(hFile)
    LINE INPUT #hFile, sLine
    IF sLine = "Online" THEN
      onofflabel.Text = "Online"
      onofflabel.ForeColor = 65280
      WaterLabel.Text = ""
      waterLabel.ForeColor = &H0
      StatusLabel.Text = ""
```

```
      StatusLabel.ForeColor = &H0
      BrewButton.Enabled = TRUE
    ELSE IF sLine = "Offline" THEN
      onofflabel.Text = "Offline"
      onofflabel.ForeColor = &HFF0000
      WaterLabel.Text = "Offline"
      waterLabel.ForeColor = &HFF0000
      StatusLabel.Text = "Offline"
      StatusLabel.ForeColor = &HFF0000
      BrewButton.Enabled = FALSE
    ENDIF
  WEND
  CLOSE #hFile
  SHELL "rm /tmp/eccheck"
END

PUBLIC SUB StatusButton_Click()
  DIM hFile AS File
  DIM sLine AS String

  SHELL "eccheck > /tmp/eccheck"

  WAIT 5
  hFile = OPEN "/tmp/eccheck" FOR READ
  WHILE NOT Eof(hFile)
    LINE INPUT #hFile, sLine
    IF sLine = "Online" THEN
      onofflabel.Text = "Online"
      onofflabel.ForeColor = 65280
      BrewButton.Enabled = TRUE
    ELSE IF sLine = "Offline" THEN
      onofflabel.Text = "Offline"
      onofflabel.ForeColor = &HFF0000
      WaterLabel.Text = "Offline"
      waterLabel.ForeColor = &HFF0000
      StatusLabel.Text = "Offline"
      StatusLabel.ForeColor = &HFF0000
      BrewButton.Enabled = FALSE
    ELSE IF sLine = "Water" THEN
      LINE INPUT #hFile, sLine
```

```
      IF sLine = "0" THEN
        WaterLabel.Text = "Empty"
        WaterLabel.ForeColor = &HFF0000
        BrewButton.Enabled = FALSE
      ELSE IF sLine = "20" THEN
        WaterLabel.Text = "2 Cups"
        WaterLabel.ForeColor = 0
        BrewButton.Enabled = TRUE
      ELSE IF sLine = "30" THEN
        WaterLabel.Text = "3 Cups"
        WaterLabel.ForeColor = 0
        BrewButton.Enabled = TRUE
      ELSE IF sLine = "40" THEN
        WaterLabel.Text = "4 Cups"
        WaterLabel.ForeColor = 0
        BrewButton.Enabled = TRUE
      ELSE
        WaterLabel.Text = "What?"
        WaterLabel.ForeColor = 0
      ENDIF
    ELSE IF sLine = "Status" THEN
      LINE INPUT #hFile, sLine
      IF sLine = "a0" THEN
        StatusLabel.Text = "Idle"
        StatusLabel.ForeColor = 0
      ELSE IF sLine = "a2" THEN
        StatusLabel.Text = "Brewing..."
        StatusLabel.ForeColor = 0
      ELSE
        StatusLabel.Text = "Idle"
        StatusLabel.ForeColor = 0
      ENDIF
    ENDIF
  WEND
  CLOSE #hFile
  SHELL "rm /tmp/eccheck"
END

PUBLIC SUB BrewButton_Click()
  SHELL "ecbrew"
```

```
  StatusLabel.Text = "Brewing..."
END
```

clicheck.c

```
#include "brew.h"

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

  if(argc != 1){
    fprintf(stderr, "Too many arguments, aborting\n");
    exit(1);
  }
  int x;

  system("sudo ifconfig eth0 192.168.0.176");
  system("arp -s 192.168.0.225 00:6F:66:66:65:65");

  for(x=0;x<2;x++){
    if(ping()){
      x--;
      printf("Offline\n");
      return 'n';
    }
  printf("Online\n");
  return 'y';
  }
}
```

clicheck.c

```
#include "brew.h"

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

  if(argc != 1){
    fprintf(stderr, "Too many arguments, aborting\n");
    exit(1);
```

```
  }
  int x;;

  system("sudo ifconfig eth0 192.168.0.176");
  system("arp -s 192.168.0.225 00:6F:66:66:65:65");

  for(x=0;x<5;x++){
    if(ping())
      x--;
  }

}
```

Installation Script
install.sh

```
#!/bin/sh
#this is my install script v2

# This section is to compile all the sources then moves the executables into cli folder for
# later
gcc -o cli/brew source/brew3.c
gcc -o cli/ecbrew source/clibrew.c
gcc -o cli/eccheck source/clicheck.c
cp cli/* /usr/bin
# This copies the GUI executable to the "~" directory of the current user (should be root,
# but sudo might place in in home)
cp GUI/EtherCoffee.gambas ~
```

PLD for the GAL

```
Name     Lab 3 ;
PartNo   G20V8A ;
Date     24/09/2009 ;
Revision 01 ;
Designer Calvin O Ference ;
Company  None ;
Assembly None ;
Location BB ;
Device   g20v8a ;
```

```
/* INPUTS */

PIN [5..11] = [A15..A09];
PIN 13 = IOM;
PIN 14 = RD;
PIN 23 = WR;


/* OUTPUTS */
PIN 15 = ROMSEL;
PIN 16 = RAMSEL;
PIN 17 = MEM_RD;
PIN 18 = MEM_WR;
PIN 19 = IORD;
PIN 20 = IOWR;
PIN 21 = IO_BLK;
PIN 22 = BLK1;

/* EQUATIONS */
FIELD ADDRESS = [A15..A09];
IORD = !IOM # RD;
IOWR = !IOM # WR;
MEM_RD = IOM # RD;
MEM_WR = IOM # WR;
IO_BLK =  A15 #  A14 #  A13 # !IOM # (RD & WR);
BLK1   = !A15 #  A14 #  A13 # IOM;
ROMSEL = !A15 # !A14 # !A13 # IOM;
RAMSEL =  A15 #  IOM;
```

Appendix C

Appendix C – Parts List		
Power Supply Unit – Appendix A – Figure 1		
Designator	Part	Value
C1, C2	Capacitor	2200µF
C3, C4	Capacitor	100µF
D1, D2, D3, D4	Diode	1N4002
D5, D6	LED	Green
F1	Fuse	15A250V
J1	Terminal Block	8 Terminal
R1	Resistor	180Ω
R2	Resistor	390Ω
S1	Power Switch	N/A
T1	Transformer	N/A
U1	Voltage Regulator	7805
U2	Voltage Regulator	7812
Switch – Appendix A – Figure 3		
K1	Relay	15 Amp 120 VAC
Q1	Transistor	2N2222
R3	Resistor	1.2kΩ
U3	Operation Amplifier	LM324
Intel 8088 Minimal System – Appendix A – Figure 5, 6, 7, 8, 9		
C5, C6, C7, C8, C9	Capacitor	10µF
R3, R4	Resistor	47kΩ
R5, R6	Resistor	1kΩ
S2	Push-Button Switch	N/A
U4	Microprocessor	Intel 8088
U5	Latch	74LS373
U6	GAL	G20V8
U7	ROM	Atmel 28C64B
U8	RAM	HM62256B
U9	3 to 8 Decoder	74HC138
U10	UART	8251
U11	Dual Driver/Receiver	MAX232
U12	12 Stage Binary ripple counter	74HC4040
U13	Programmable Peripheral Interface	8255
U14	Programmable Interrupt Controller	8259
U15	Supply and Control Voltage Regulator	8284
U16	Ethernet Controller	CS8900A-H
U17	Buffer	74HC245
U18	Analog to Digital Converter	ADC0804
Y1	Crystal	14.9MHz

Appendix D

Appendix D – Bill of Materials					
Part	**Part Number**	**Quantity**	**Price**		**Total Price**
Power Supply					
100 µF Capacitor	100R50	2	0.21		0.42
2200 µF Capacitor	2200R50	2	0.74		1.48
Diodes	1N4002	4	0.06		0.24
Heat Sink	HS-310	2	0.29		0.58
LED	Green	2	0.10		0.20
Resistor	N/A	2	0.04		0.08
Transformer	TRF-310	1	17.99		17.99
5V Voltage Regulator	7805	1	0.42		0.42
12V Voltage Regulator	7812	1	0.42		0.42
Switch					
Resistor	N/A	1	0.04		0.04
Relay	N/A	1	4.99		4.99
Operation-Amplifier	LM324N	1	0.35		0.35
Transistor	2N2222	1	0.37		0.37
Micro Controller					
12 Stage Binary Ripple	74HC4040	1	0.71		0.71
3 to 8 Decoder	74HC138	1	0.35		0.35
10 µF Capacitors	10R50V	5	0.09		0.45
Analog to Digital Converter	ADC0804	1	3.49		3.49
Buffer	74LS245	1	0.49		0.49
Crystal	14.74MHz	1	2.89		2.89
Dual Driver/Receiver	MAX232	1	2.49		2.49
Ethernet Controller	CS8900A-H	1	41.95		41.95
GAL	GAL20V8B	1	2.49		2.49
HEX Display	TIL311	1	12.99		12.99
IR Sensor	GP2D120	1	8.99		8.99
Latch	74LS373	1	0.49		0.49
Microprocessor	8088	1	2.89		2.89
Programmable Communication Interface	8251	1	2.89		2.89
Programmable Interrupt Controller	8259	1	2.79		2.79
Programmable Parallel Interface	8255	1	2.99		2.99
Push-Button Switch	PBS-190	1	0.69		0.69
Random Access Memory	HM62256B	1	6.99		6.99
Read Only Memory	AT28HC64B	1	4.99		4.99
Resistors	N/A	4	0.04		0.16
16-Pin Wire-wrap socket	MS16WW	3	1.12		3.36
18-Pin Wire-wrap socket	MS18WW	1	1.19		1.19
20-Pin Wire-wrap socket	MS20WW	3	1.39		4.17
24-Pin Wire-wrap socket	MS24WW	1	1.49		1.49
28-Pin Wire-wrap socket	MS28WW	4	1.72		6.88
40-Pin Wire-wrap socket	MS41WW	3	2.69		8.07
Miscellaneous					
Cable	N/A	1	19.99		19.99
Coffee Machine	DCM500	1	19.99		19.99
Fans	N/A	2	2.99		5.98
Fuse	15A	1	1.99		1.99
Fuse Holder	FH110	1	0.89		0.89
DPST Switch	AT Power-Supply	1	2.83		2.83
Metal Rods	N/A	1	2.99		2.99
Nuts and Bolts	N/A	48	0.09		4.32
Terminal Block	N/A	1	4.99		4.99
Vector Boards	N/A	2	2.49		4.98
Wood	N/A	1	11.99		11.99
Total					235.85

Special Acknowledgment to...

My Father – For all the cooking and putting up with my [insert appropriate profanity], and all the mechanical advice you offered.

Mr. Plaitis – For helping me find a job and keeping me on track with my education.

Mr. Markou – For all the help you offered me for my project and for all the knowledge and the cool technical stories you shared.

Mr. Lavoie – For the knowledge you shared and for being a fun person to talk with.

Mr. Lorkovich – For the cool and pleasant class time you offered in both of my first and last semester.

Mike – For all the help and running around you did. Greatly appreciated.

Andrea – For all the moral support, the gaming time and for checking in with my mental sanity. And all the hugs of course.

Charles – For all the support and the drinks you offered from your minibar.

Daniel – For all the fascinating philosophical/theological talks we had and all the other things you brought with your cool South African accent.

Etienne – For being yourself bud.

Sam Lee – For all the video game talk we had; most fun.

Marie-Claude, Samuel, Stephanie, Vanessa, Vicky and the rest of the gang – For all the moral support and all the parties we had together. Thanks guys!