Project 003070

A Green Solution to Basement Humidity Control

Eligible Part: WIZNet W7100

Much of my hobby equipment resides in my basement. It's an unfinished basement and doesn't have any significant water leaks but it gets very humid during warm months due to moisture penetration through the walls and floor. This humidity can inflict serious rust damage on the equipment and can cause mold growth on books and magazines. I have been running a dehumidifier during most of the year but it has a fairly significant current draw and the electricity bill can add up. One of my many projects was a temperature/humidity display that monitored the indoor and outdoor conditions. It occurred to me that armed with the information from the measurement device it should be possible to determine when the outside air is drier than the basement air and use ventilation rather than the dehumidifier to dry out the basement. With this idea sprung forth the basement humidity control project.

The basic concept of the project is to calculate the Water Vapor Pressure from the temperature and humidity readings and, in those cases where the outside air is drier, enable a ventilation system rather than the dehumidifier. The elaboration of this basic idea was to add a memory stick to record all the data, a local display to indicate conditions and status, and a remote interface over the Ethernet made possible by the WIZNet W7100. This remote interface allows a browser to display all of the conditions and device status as well as to take control of the system and manually enable or disable the devices. It also allows the file data stored on the memory stick to be transferred to the PC without having to unplug the memory and manually download the data to a PC. An additional pair of humidity/temperature sensors monitors the inlet and outlet of my heating/air conditioning system. This is not part of the humidity control but it is interesting information to have available.

I am an assembly language programmer, so my first task was to translate the kick-start information supplied by WIZNet into an assembly language equivalent. Rather than just transferring the assembly code generated by the compiler, I translated this code to my "style". This part of the project was not without bumps but it was a great experience to figure out just how this very capable chip worked. Once having arrived at this baseline I then began adding some of my familiar interfaces to the WIZNet W7100. My temperature/humidity sensors are I2C devices but the W7100 lacks this type of interface. Adding this interface was easily accomplished with a few short "bit banging" routines. The W7100 uses weak pull-ups on pins so it was easy to simulate the open source pin normally used for the I2C Data (SDA). A little time spent tuning the timing and this interface sprung to life. The interface to the memory stick using the V-Drive2 device did not go as smoothly. In fact, I learned a lot more about this interface than I had expected.

The V-Drive2 device allows a user to talk to it over a serial or an SDI interface. I wanted to leave the serial lines available on the W7100 so I elected to use the SDI mode. This looked simple, but without an active pull-up on the W7100 port pins the rising edge of the clock signal was too slow for the V-Drive. It kept getting confused and refused to talk. Once I tracked down this problem it was easy (if not pleasing) to add a Schmitt trigger IC (74HCT14) and clean up the clock edge. Since the '14 is a hex package I went ahead and added inverters to all of the lines going to the V-Drive. This wasn't necessary but it was a bit of insurance.

Packaging was the next challenge. The evaluation board for the W7100 is very well equipped with interfaces and pins so I decided to use it without modification and just add my extra parts to the board. It was a tight squeeze. I added the 74HCT14, a DS1307 Real Time Clock with battery and crystal, an 82B715 I2C extender chip, and an assortment of transistors, resistors, capacitors, and connectors. There is not much room left but it all fit and worked nicely. I added my own 5-volt regulator so I could bring 12 volts to the assembly for the relays, LCD backlight, and distribution to the remote I2C devices. All of this was packaged in a fairly small case with the LCD display on the front and the V-Drive and external connectors for the I2C, the relay control box, and power connector on the back. A push button switch is also included to signal the W7100 the user wants to suspend disk operations to change out the USB device.

The control of the vent, fan, and dehumidifier is accomplished by a set of relays mounted in a junction box and driven by transistors on the evaluation board. The remaining three gates in the 74HCT14 were used to drive these transistors. This added additional isolation from the switching transients in the relays.

The software development went fairly smoothly. The main processing loop is responsible for reading the I2C devices, checking for Ethernet requests, and writing to the LCD display and the memory stick. The interface to the temperature/humidity data is ASCII so it would be easy for other devices and interfaces to be substituted without affecting much of the system. The basic logic reads the indoor and outdoor conditions and then calculates the water vapor pressure from a table look up. It compares these two values and decides to open or close a vent, turn a fan off or on, or enable the dehumidifier. The data is displayed on the LCD along with the status of the vent, fan, and dehumidifier.

The W7100 acts as a web server to a browser to display three panels with the conditions (temperature/humidity/water vapor pressure), status (vent open/closed, fan off/on, dehumidifier enabled/disabled), and control/data transfer. The control and data transfer panel allows the operator to assume manual control of the system, to set the DS1307 real time clock, and most importantly to select files for transfer to the PC. All files are in ASCII so they can be imported into Excel for plotting and analysis. The data is recorded as 256 bytes once every 4 seconds and a file of over 5 megabytes is generated each day. I also included an FTP protocol so the device can be addressed as an FTP server and it will return directories and files to the client.

The remote I2C sensors use a single chip computer (89LPC925) with a TLC555 oscillator tuned by a capacitive humidity sensor and a “1-wire” thermometer chip for the temperature. A crystal was used on the microprocessor to provide stability for the time measurement of the 555 output. The units use RJ-11 connectors and can be daisy chained or driven in a star configuration. The unit at maximum distance from the W7100 control processor also uses the 82B715 I2C extender but the other units drive and read the I2C lines through the microprocessor directly. The temperature and humidity data is stored in fixed locations in memory and the W7100 reads these cells to update its status.

All pieces work nicely together and the W7100 made this a fairly easy task to switch from the full time dehumidifier to an intelligent humidity management system for my basement.

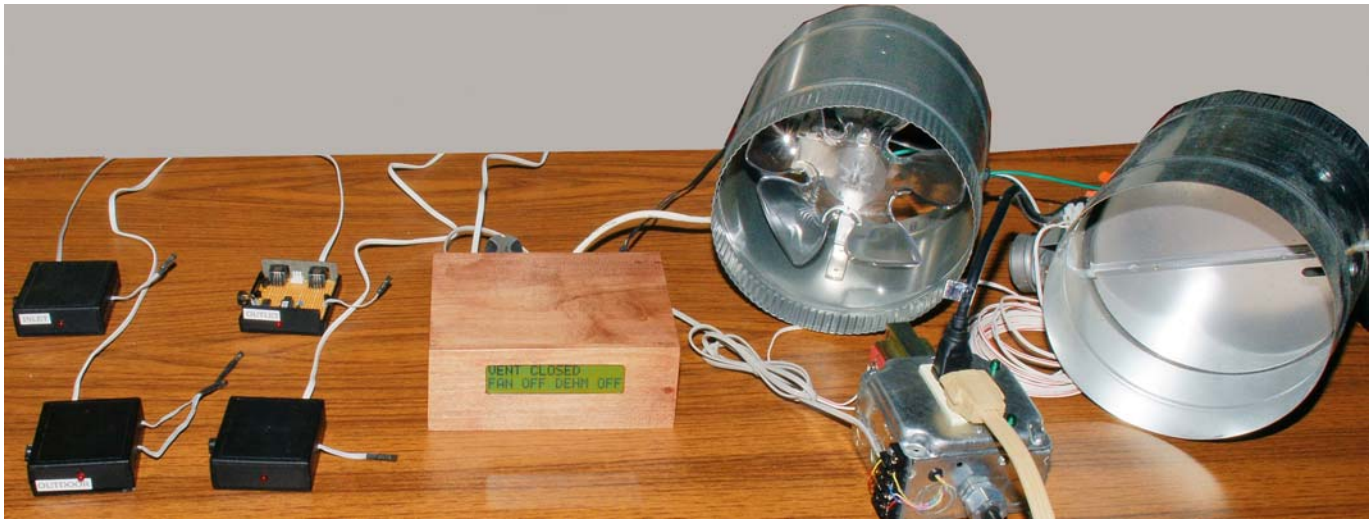


Photo 1 – Complete Assembly on desktop

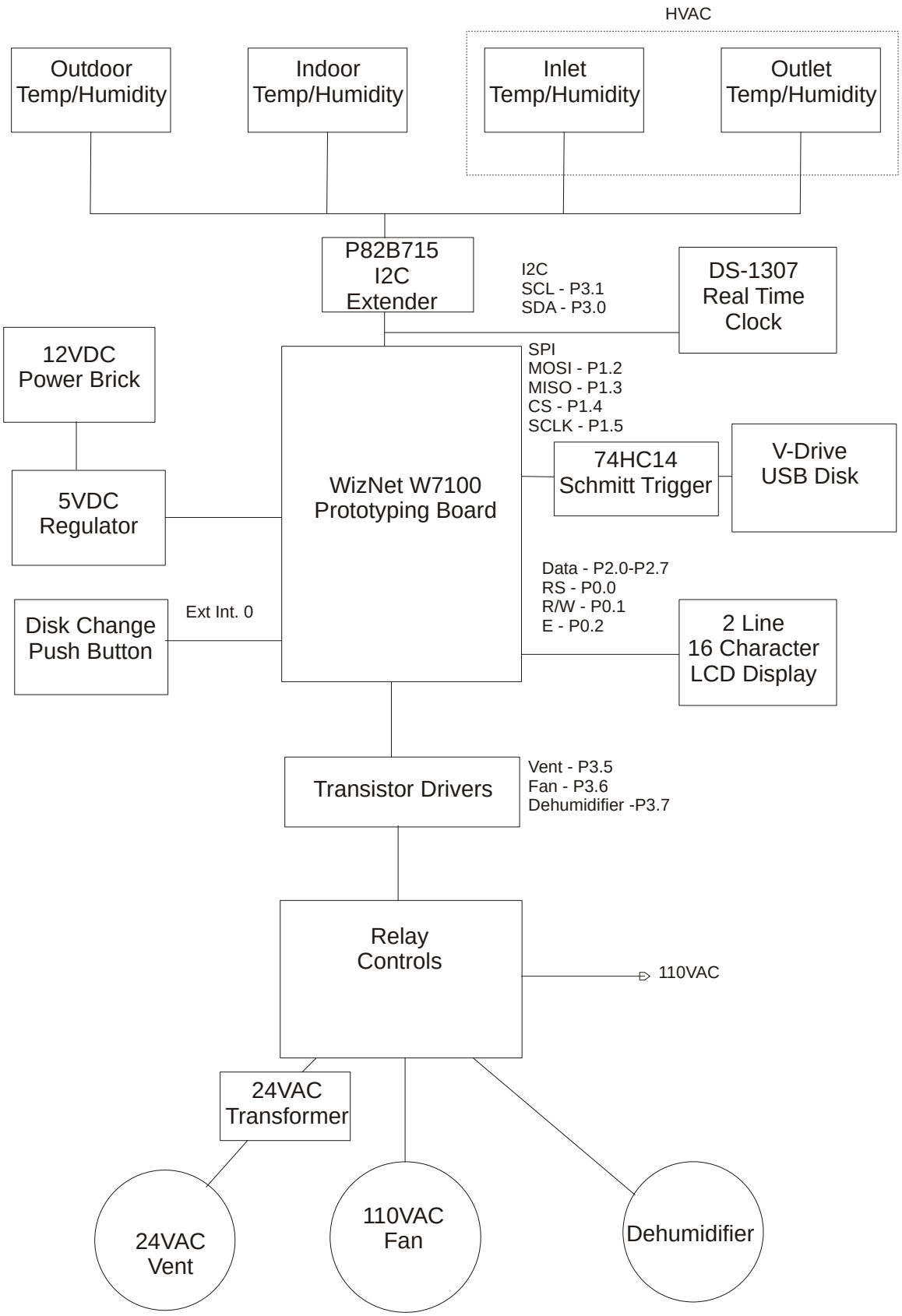
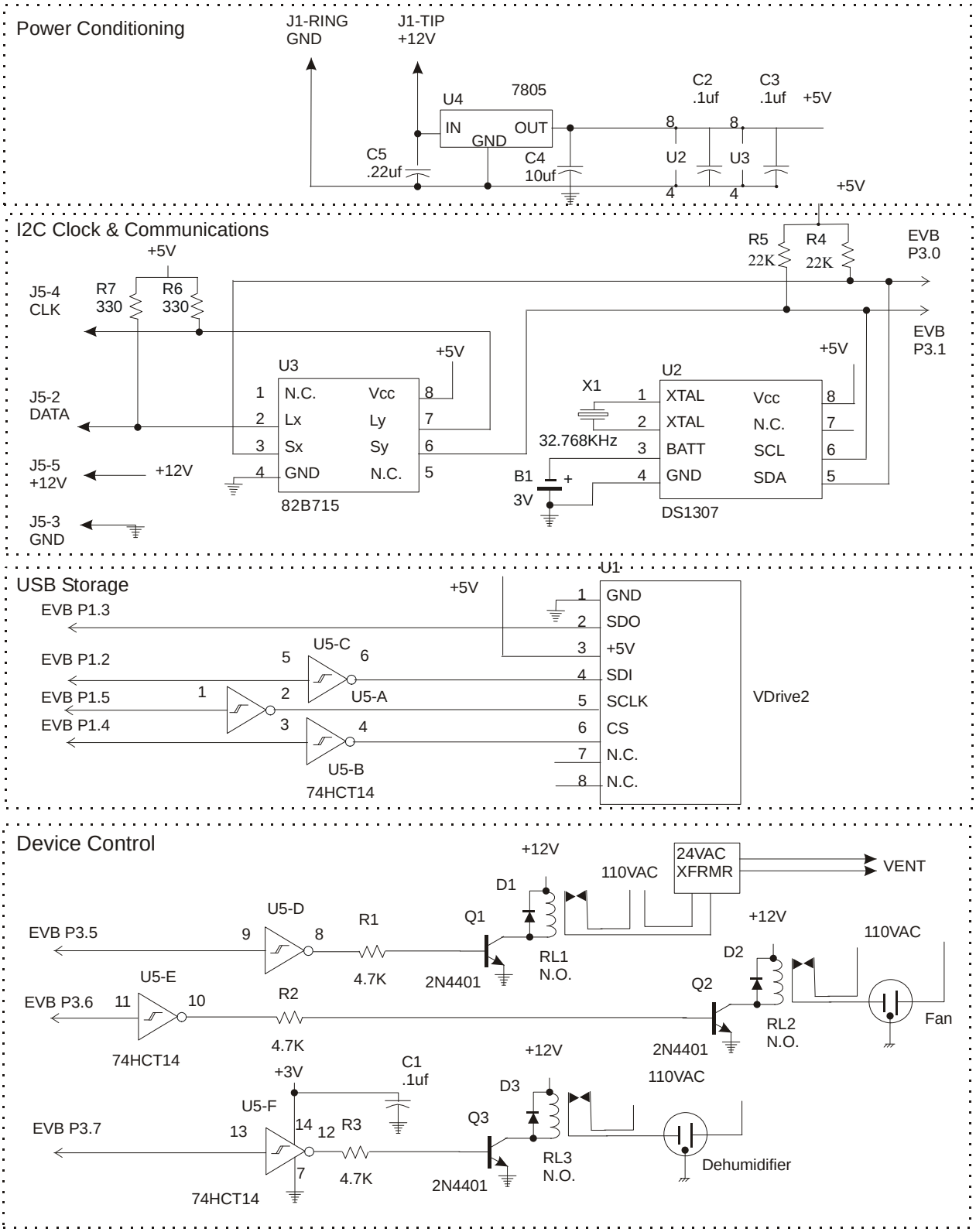
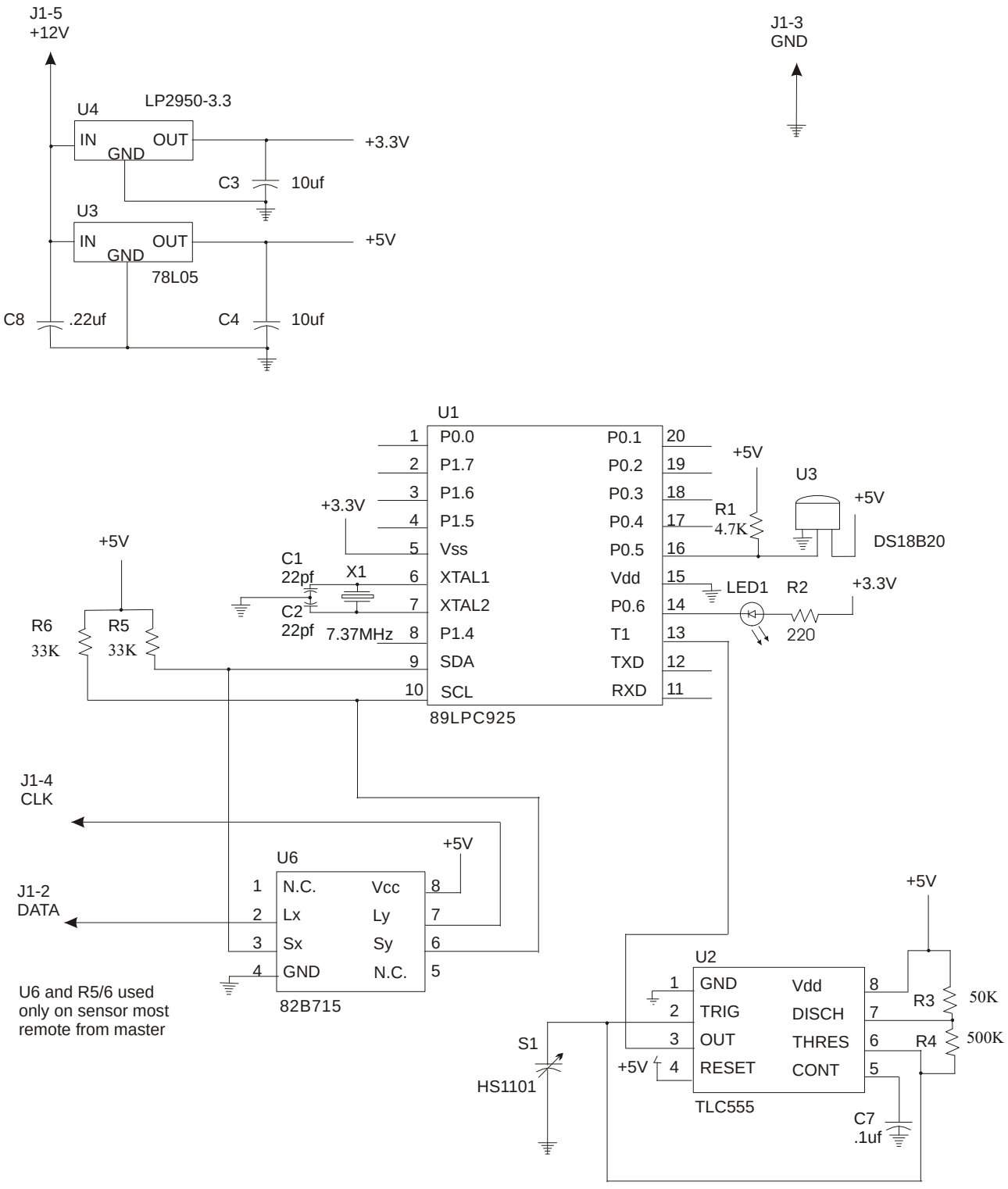


Figure 1 - System Block Diagram



System Schematic - Basement Humidity Control
Project 003070

Figure 2 - System Schematic



Humidity Sensor (1 of 4) for Basement Humidity Control
Project 003070

Figure 3 - Sensor Schematic

Figure 4 - Sample Code - Main Processing Loop

```
IT_00:
        CLR     LCD_ON
        CALL    NETWORK          ; Check for network activity
        JB      RETR,IT_00       ; Suspend other activity if file xfer
        SETB    LCD_ON
        JB      EX0,IT_01
        CALL    CHANGE_DSK       ; Interrupt disabled = button pressed
IT_01:
        JNB     ONE_SEC,IT_00
        MOV     WATCH,#15        ; 15 seconds max revisit
        CLR     ONE_SEC
        CALL    LINES1_8         ; 1/second refresh buffer
        CALL    LCD_DISPLAY      ; Output to LCD display
        INC     OUT_TIMER
        MOV     A,OUT_TIMER
        CJNE    A,#4,IT_00       ; 4 seconds write to V_DRIVE/change display
        MOV     OUT_TIMER,#0
        INC     V_DRIVE_CTR
        ANL     V_DRIVE_CTR,#1
        INC     DISPLAY_SELECT
        ANL     DISPLAY_SELECT,#3
;  Read clock
        MOV     R0,#CMEM_TIME    ; Read clock chip
        MOV     R1,#T_ARRAY
        MOV     R2,#9            ; 9th word is DST
        CALL    READ_CMEM
IF V_DRIVE
; Check if file needs to be opened
        MOV     A,V_DRIVE_CTR
        JZ      MI_06            ; Jump if open
; Open file for write
        CALL    OPEN_FILE
MI_06:
; Start the write operation for 256 characters +cr+lf
; (21+1) is current time
; (16+1+16+1) are the outside-inside, inlet-exhaust temp/hum readings
; (16+1+16+1) are the inlet-exhaust temp/hum readings
; (16+1+16+1) are the outside-inside water vapor pressure readings
; (16+1+16+1) are the vent+fan/dehumidifier status
; 256-158 = 96 blank filled then cr and lf
        CALL    PRINT_V
        DB      'WRF ',0
        MOV     A,#0             ; MSB
        CALL    V_OUT
        MOV     A,#0
        CALL    V_OUT
        MOV     A,#01            ; 100H =256
        CALL    V_OUT
        MOV     A,#00
        CALL    V_OUT
        MOV     A,#CR
        CALL    V_OUT
; Print 21+1 characters of current time
        CALL    TIME_TO_USB
; Output all readings to V_DRIVE
```

```
        MOV     DPTR,#MEM_BUFFER
; Output 8-16 character lines each followed by a blank
        MOV     R7,#8
MI_15:
        MOV     R6,#16
MI_20:
        MOVX    A,@DPTR
        CJNE    A,#0DFH,MI_21     ; Substitute printable character for degree sign
        MOV     A,#0B0H
MI_21:
        CALL    V_OUT
        INC     DPTR
        DJNZ    R6,MI_20
        MOV     A,#' '
        CALL    V_OUT
        DJNZ    R7,MI_15
; Output 96 characters of blanks
        MOV     R7,#96
MI_40:
        MOV     A,#' '
        CALL    V_OUT
        DJNZ    R7,MI_40
; Terminate line with CR/LF
        MOV     A,#CR
        CALL    V_OUT
        MOV     A,#LF
        CALL    V_OUT
; Total write = 6*(16+1+16+1)+50+2 = 256
        CALL    GET_PROMPT        ; Conclude the write operation
; Close file after 'n' writes
        MOV     A,V_DRIVE_CTR
        JZ      MI_50
        JMP     IT_00
MI_50:
        CALL    PRINT_V
        DB      'CLF ',0
        CALL    OPN_CLS
; If request is pending to remove unit then indicate it can be changed
        JB      WRITE_PAUSE,MI_60
        JMP     IT_00
MI_60:
        CLR     LOCK_DISPLAY
        MOV     WATCH,#0          ; Turn off watchdog
        SETB    LCD_ON
        CLR     MEM_ON
        CLR     SER_ON
        CALL    RC_DISPLAY
        DB      1,1,'Insert new Drive',0
        CALL    RC_DISPLAY
        DB      2,1,'Pgm will restart',0
        CLR     WRITE_PAUSE
        CALL    GET_PROMPT
        CLR     LOCK_DISPLAY
        CLR     IE0
        SETB    EX0               ; Reenable the push button logic
        JMP     IT_00
```