Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 6 Issue 12, December - 2019

A Microcontroller Based Commercial Storage Temperature Monitor: Design and Construction

John Shipley
School of Undergraduate Studies
Excelsior College, Albany, NY 12203, USA
jhship@yahoo.com

Sohail Anwar
Division of Business, Engineering, and Information System Technology
Penn State University, Altoona College Altoona, PA 16601, USA
sxa15@psu.edu

Patrick Favier
Department of Electrical Engineering
IUT Bethune, Universite d' Artois, Bethune, France
Patrick.favier@univ-artois.fr

***Abstract* – restaurants have to comply with state laws and federal regulations that require hourly monitoring of the restaurant food storage containers. Temperature monitoring is generally done manually which sometimes requires the use of dedicated employees. A typical food establishment may have eight to twelve food storage units that require hourly temperature monitoring. Manually recording the temperature and tracking measurements is intensive and time consuming. Keeping historical records of these manual temperature measurements is not easy and requires a lot of effort. This manuscript presents the design and construction details for a low cost central wireless micro controller-based temperature monitoring system. This system is self-monitoring to an acceptable level. It provides the necessary control data and creates historical records for analysis.**

Keywords—storage; temperature monitoring; microcontroller; control

1. INTRODUCTION

In food industry, monitoring food storage temperature data is essential. In many restaurants and other food establishments, the temperature monitoring of food storage units is done manually. The manual temperature monitoring and recording is usually time intensive and requires significant effort.

As described in [1], automated temperature monitoring and control is a viable alternative for manual temperature monitoring. Automated temperature monitoring helps ensure food safety during processing and storage. It is also effective in meeting the requirements for electronic record keeping. It is suggested in [1] that wireless systems may serve as a useful tool for temperature monitoring in food storage facilities. At present, wireless systems can allow for four to six temperature sensors, increasing the monitoring system accuracy and reducing the installation cost significantly. An automated temperature monitoring system based on Internet of Things (IOT) is proposed in [2]. Basically, the concept of IOT involves connecting any device with an ON and OFF switch to the Internet. The Internet of Things is a system of physical devices, appliances, and other things supported by hardware, software, sensors, and Internet. In [2], the design of an automated system which performs the real-time monitoring of temperature and humidity through Internet is described. This system uses Node MCU micro controller board and a DHT-11 sensor.

The low cost microcontroller based food storage automated temperature monitoring system described in this manuscript exhibits a good accuracy, provides the necessary control data, and generates historical data records for analysis.

2. SYSTEM OVERVIEW

The automated temperature monitoring system described in this manuscript comprises a wireless main-console and remote sensor unit utilizing the IEEE 802.15.4 wireless personal area network protocol. The characteristic behavior of this type of network entails very short bursts of message traffic over short distances. More information regarding IEEE 802.15.4 is provided in [3].

The main console of this temperature monitoring system can monitor and record up to twelve (12) remote sensor controllers. The main console includes a power supply, the processor board, a 4-line 20-character LCD module, and a wireless transceiver. The processor used for this automated temperature monitoring system is the Microchip PIC16F870 microcontroller. The PIC16F870 features 64 bytes of EEPROM data memory, self-programming, five channels of 10-bit analog-to-digital (A/D) converter, and a Universal Asynchronous Receiver Transmitter (USART). All these features make PIF16F870 microcontroller very well suited for advanced level analog-to-digital applications in industrial and consumer electronic systems. More information pertaining to this microcontroller is provided in [4]. The LCD module used in this system is any 4-line 20-character/line LCD with back-lighting similar to the MTC-20400 B LCD module. The transceiver used for this temperature monitoring system is the XBEE XB24-AC1. The indoor range is 50 feet. External electric power is regulated through an LM 7805 TO220 package and it can be used with any regulated 7-15V DC power supply.

The remote sensor unit includes a power supply, a DS18B20 digital temperature sensor, and a wireless transceiver. Just like the main console, the remote sensor unit also uses the PIC16F870 microcontroller. The transceiver is the XB24-ACI module.

3. DESIGN AND SOFTWARE

The design schematic for the main console of the PIC16F870 microcontroller based temperature monitoring system is provided in Figure 1. The design schematic for the remote sensor unit is shown in Figure 2. The reason for the use of PIC16F870 microcontroller is its low cost and availability. The accuracy of the DS18B20 digital temperature sensor is ± 0.5°C in the temperature range of -10°C to +85°C. More information regarding this sensor is provided in [5]. The Hitachi HD44780U controller driver is used for this temperature monitoring system. A single HD44780U can display up to one 8-character line or two 8-character lines. It can be configured to drive a dot-matrix LCD under the control of a 4 or 8-bit microprocessor. More information regarding HD44780U can be found in [6].

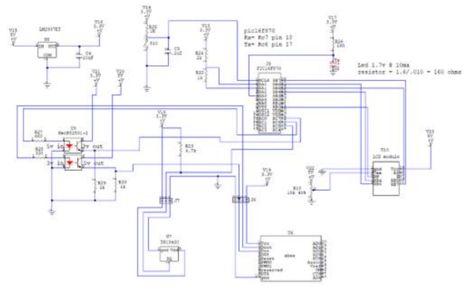

Fig. 1: The schematic for main console

The software for the temperature monitoring function is partially listed in the APPENDIX. The code is compiled using PICBASIC PRO Compiler. The target PIC microcontroller for this code is a 28-pin PIC16F870.

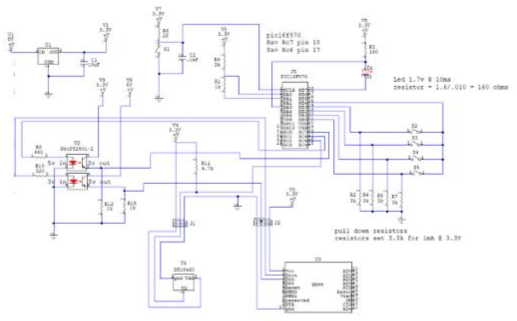

Fig. 2: The schematic for the remote sensor

4. CONCLUSION

This manuscript presents the design and construction details for a PIC16F870 microcontroller based temperature monitoring system for food storage units. This low cost ($50-$100) system uses XBEEXB24 – AC1 wireless transceiver. This system helps ensure food safety during food processing and storage. The system provides the necessary control data and generates historical data records pertaining to temperature monitoring.

REFERENCES

[]] Automated temperature monitoring and control ensure food safety https://www.foodengineeringmag.com/articles/96675_automated_temperature_monitoring_and_control_ensure_food_safety?

[2] A. Karim, M. Hassan, M. Akanda and A. Mallik, "Monitoring Food Storage humidity and temperature data using IOT," MOJ Food Processing & Technology, 6(4), pp. 400-404, 2018.

[3] IEEE 802.15.4_2015_IEEE Standard for Low-Rate Wireless Networks. https://standards.ieee.org/standard/802_15_4-2015.html

[4] PIC16F870_Microcontrollers and Processors https://www.microchip.com/wwwproducts/en/PIC16F870

[5] DS18B20_Programmable Resolution 1-Wire Digital Thermometer https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[6] HD44780U – Dot Matrix Liquid Crystal Display Controller/Driver https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

APPENDIX

```
    '****************************************************************
'*  Name    : xbee temp monitor.pbp                   *
'*  Author  : JS                                *
'*  Notice  :                                 *
'*          :                               *
'*  Date    : 7/06/2019                          *
'*  Version : 1.0                              *
'*  Notes   : pic16f870                          *
'*          : DC-20mhz, 2Kx14 flash program memory          *
'*          : 128x8 bytes data memory, 64x8 bytes eeprom memory *
'****************************************************************

' Name        : xbee temp monitor.pbp
' Compiler    : PICBASIC PRO Compiler 2.6
' Assembler   : PM or MPASM
' Target PIC  : 28-pin 16f870
' Hardware    :
' Oscillator  : 4MHz external
' Description : remote temperature sensor Maxim DS18B20

  define osc 4

'port setup
  trisa = %00111111
  trisb = 0
  trisc = %10000000    'rc7 is rs232 receive, set as input

'Define LCD registers and bits
   DEFINE LCD_RSREG PORTB ' Set LCD Register Select port
   DEFINE LCD_RSBIT 1 ' Set LCD Register Select bit
   DEFINE LCD_RWREG PORTB
   DEFINE LCD_RWBIT 0
   DEFINE LCD_EREG PORTB ' Set LCD Enable port
   DEFINE LCD_EBIT 2 ' Set LCD Enable bit
   DEFINE LCD_BITS 4 ' Set LCD bus size (4 or 8 bits)
   DEFINE LCD_LINES 4 ' Set number of lines on LCD
   DEFINE LCD_DREG PORTB ' Set LCD Data port
   DEFINE LCD_DBIT 4 ' Set starting Data bit (0 or 4) if 4-bit bus
   DEFINE LCD_COMMANDUS 1500 ' Set command delay time in us
   DEFINE LCD_DATAUS 44 ' Set data delay time in us

'uart setup
   'rw-0=readwrite default=0, r-0=readonly(x=unknown), u-1=not implimented
   txsta.0 = 0 'rw-0,tx9d=9th bit of transmit data
   'txsta.1 = 0 'r-1,trmt=transmit shift register status bit
   txsta.2 = 1 'rw-0,brgh=high baud rate select bit
   'txsta.3 = 0 'unimplemented
   txsta.4 = 0 'rw-0,sync=usart mode select bit
   txsta.5 = 1 'rw-0,txen=transmit enable bit
   txsta.6 = 0 'rw-0,tx9=bit transmit enable bit
```

```
    txsta.7 = 0 'rw-0,csrc=clock source select bit

    'rcsta.0 = 0 'r-x,rx9d=9th bit received data
    'rcsta.1 = 0 'r-0,oerr=overrun error bit
    'rcsta.2 = 0 'r-0,ferr=framing error bit
    'rcsta.3 = 0 'u-0,not implemented
    rcsta.4 = 1 'rw-0,cren=continuous receive enable bit
    rcsta.5 = 0 'rw-0,sren=single receive enable bit
    rcsta.6 = 0 'rw-0,rx9=9bit receive enable bit
    rcsta.7 = 1 'rw-0,spen=serial port enable bit

    spbrg = 25              '4mhz clk + brgh=1 then 25 = 9600 baud

'A/D setup
    'Define ADCIN parameters
    'Define ADC_BITS     8   ' Set number of bits in result
    'Define ADC_CLOCK    3   ' Set clock source (3=rc)
    'Define ADC_SAMPLEUS 50  ' Set sampling time in uS

   'ADCON0.2=1 starts A/D conversion
   ADCON0 = %11000001      'CONVERSION CLK=Frc, Chan select=A0, A/D on
   ADCON1 = $81            'A3=VREF+, VREF-=VSS, A0=ANALOG INPUT, right justified

'set variables, ram 128 bytes
    led     var portb.3
    buz     var portc.3         ' buzzer, pwm pin
    batsel  var portc.4         ' battery select, low to read battery voltage
    tdata   Var portc.5         ' One-wire data pin
    Ctemp   var word            ' calculated temperature oC
    Ftemp   var word            ' calculated temperature oF
    tcount  var word            ' temp count from ds1820
    bat     var word
    I       var byte
    K       var byte            ' counter
    adr     var byte            ' address in decimal
    csign   var bit             ' oC, 1= negative sign, 0= no sign
    fsign   var bit             ' oF, 1= negative sign, 0= no sign
    rdata   var byte[11]        ' remote data, 5 bytes from each sensor
    chtemp  var word[16]        ' channel temperature reading
    chbat   var word[16]        ' channel battery voltage
    cntr1   var word            ' counter #1
    tries   var byte            ' number of tries

'test
    led=1   'led off
    gosub buzzer
    freqout led,4000,2  'send 1 hz for 3 seconds
    pause 5000
    led=1  'led off

main:
    'temperature and bat voltage data is saved in its raw form in chtemp[adr] and chbat[adr]
    'this is the counts from ds1820 and ADC
    'each time used, have to convert to oF using caltemp, bat voltage is used as count
    'get data
        adr=0   '0=address for console
           gosub readtemp  'read console ds1820 into chtemp[adr]
           gosub readbat   'read console battery into chbat[adr] 21ms
        for  adr= 1 to 15   'get data from sensors
             tries=3 'number tries
             led=0
             gosub getdata    'getdata sends adr and waits for response 50ms/adr
             led=1
        next adr
        gosub disptemp   'displays Ftemp from caltemp
        for I= 1 to 50 'wait 1 minute between readings
           pause 100
        next I
goto main

end
'-----------------------------------------------------------------------------
buzzer:
     freqout buz,100,4096  'send 4096 hz for 250ms
     pause 100
return
```