```
/*

Fan Speed Controller - Author Glen Popiel - KW5GP

Based on OneWire Library - Copyright (c) 2007, Jim Studt  (original old
version - many contributors since)

The latest version of this library may be found at:
  http://www.pjrc.com/teensy/td_libs_OneWire.html

OneWire has been maintained by Paul Stoffregen (paul@pjrc.com) since
January 2010.  At the time, it was in need of many bug fixes, but had
been abandoned the original author (Jim Studt).  None of the known
contributors were interested in maintaining OneWire.  Paul typically
works on OneWire every 6 to 12 months.  Patches usually wait that
long.  If anyone is interested in more actively maintaining OneWire,
please contact Paul.

Version 2.2:
  Teensy 3.0 compatibility, Paul Stoffregen, paul@pjrc.com
  Arduino Due compatibility,
http://arduino.cc/forum/index.php?topic=141030
  Fix DS18B20 example negative temperature
  Fix DS18B20 example's low res modes, Ken Butcher
  Improve reset timing, Mark Tillotson
  Add const qualifiers, Bertrik Sikken
  Add initial value input to crc16, Bertrik Sikken
  Add target_search() function, Scott Roberts

Version 2.1:
  Arduino 1.0 compatibility, Paul Stoffregen
  Improve temperature example, Paul Stoffregen
  DS250x_PROM example, Guillermo Lovato
  PIC32 (chipKit) compatibility, Jason Dangel, dangel.jason AT gmail.com
  Improvements from Glenn Trewitt:
  - crc16() now works
  - check_crc16() does all of calculation/checking work.
  - Added read_bytes() and write_bytes(), to reduce tedious loops.
  - Added ds2408 example.
  Delete very old, out-of-date readme file (info is here)

Version 2.0: Modifications by Paul Stoffregen, January 2010:
http://www.pjrc.com/teensy/td_libs_OneWire.html
  Search fix from Robin James
    http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1238032295/27#27
  Use direct optimized I/O in all cases
  Disable interrupts during timing critical sections
    (this solves many random communication errors)
  Disable interrupts during read-modify-write I/O
  Reduce RAM consumption by eliminating unnecessary
    variables and trimming many to 8 bits
  Optimize both crc8 - table version moved to flash

Modified to work with larger numbers of devices - avoids loop.
```

Tested in Arduino 11 alpha with 12 sensors.
26 Sept 2008 -- Robin James
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1238032295/27#27

Updated to work with arduino-0008 and to include skip() as of
2007/07/06. --RJL20

Modified to calculate the 8-bit CRC directly, avoiding the need for
the 256-byte lookup table to be loaded in RAM. Tested in arduino-0010
-- Tom Pollard, Jan 23, 2008

Jim Studt's original library was modified by Josh Larios.

Tom Pollard, pollard@alum.mit.edu, contributed around May 20, 2008

Much of the code was inspired by Derek Yerger's code, though I don't
think much of that remains. In any event that was..
    (copyleft) 2006 by Derek Yerger - Free to distribute freely.

The CRC code was excerpted and inspired by the Dallas Semiconductor
sample code bearing this copyright.

```
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be
included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM,
DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//-------------------------------------------------------------------------
---
*/

#include <OneWire.h>   // Include the OneWire Library

#define temp_sensor 10  // Temperature Sensor attached to Pin 10
#define red 7  // Red LED on pin 7
#define blue 8  // Blue LED on pin 8
#define green 9  // Green LED on pin 9
#define fan 6  // Fan drive on pin 6
#define alarm 5  // alarm buzzer on pin 5
#define low_temp 80  // temperature to start fan
#define high_temp 95  //  temperature to turn fan on full
#define alarm_temp 110  // alarm temperature

int DS18S20_Pin = temp_sensor; //DS18S20 Signal pin on temp_sensor pin

//Temperature chip I/O
OneWire ds(DS18S20_Pin);  // on temp_sensor pin

void setup(void)
{
  for (int x = 5; x <=9; x++)  // Set the Pin Modes for Pins 5 through 9
  {
    pinMode(x, OUTPUT);
    if (x <= 6)  // For pins 5 and 6 start Low, 7-9 start High (LED off)
    {
      digitalWrite(x, LOW);
    } else {
      digitalWrite(x, HIGH);
    }
  }
```

```
  ledOff();  // Turn LED Off if below low_temp
}

void loop(void)
{

  float temperature = getTemp();  // Read the Temperature Sensor
  if (temperature < low_temp)  //  Everything off if below low_temp
  {
    ledOff();
    digitalWrite(fan, LOW);
    digitalWrite(alarm, LOW);
  }
  if (temperature >= low_temp && temperature <= high_temp)  // Run the
Fan as a proportion of the temp
  {
    analogWrite(fan, map(temperature,low_temp, high_temp,30,255));  //
Starting PWM should be 30 or above to prevent fan stall
    ledBlue();  // Indicate Fan Running - Temp within range
    // Turn on the fan at slow speed
  }
  if (temperature > high_temp && temperature < alarm_temp)  // Turn the
Fan on full speed
  {
    analogWrite(fan,255);  // Set the Fan to Max Speed
    ledGreen();  // Indicate Fan Running - Max Speed
  }
  if (temperature >= alarm_temp)  // Overtemp Alarm
  {
    ledRed();  // Indicate Temp over Limit
    digitalWrite(alarm,HIGH);  // Sound the Alarm
  } else {
    digitalWrite(alarm, LOW);
  }
}

//  ------------------------ Functions ----------------------------------
----

float getTemp()   //returns the temperature from one DS18S20 in DEG
Farenheit
{
  byte data[12];
  byte addr[8];

  if ( !ds.search(addr)) //  Look for more sensors
  {
      //no more sensors on chain, reset search
      ds.reset_search();
      return -1000;
  }

  if ( OneWire::crc8( addr, 7) != addr[7])   // Check the CRC
  {
```

```
      return -1000;
  }

  if ( addr[0] != 0x10 && addr[0] != 0x28) // Verify the Device type
  {
      return -1000;
  }

  ds.reset();
  ds.select(addr);
  //  ds.write(0x44,1); // start conversion, with parasite power on at
the end
  ds.write(0x44); // start conversion, without parasite power on at the
end

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE); // Read Scratchpad


  for (int i = 0; i < 9; i++)
  { // we need 9 bytes
    data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];  // Split the data into MSB and LSB
  byte LSB = data[0];

  float tempRead = ((MSB << 8) | LSB); //using two's compliment
  float Centigrade = tempRead / 16;  // Calculate temperature in
Centigrade
  float Farenheit = Centigrade * 1.8 + 32.0;  // Convert from Centigrade
to Farenheit

  return Farenheit;  // Return Farenheit - Can be changed to return
Centigrade if desired

}


// LED Off function
void ledOff()
{
  digitalWrite(red, HIGH);  // Set all RBG LED pins High (Off)
  digitalWrite(green, HIGH);
  digitalWrite(blue, HIGH);
}

// RED LED ON function
void ledRed()
{
  digitalWrite(red, LOW);  // Turn on the RGB Red LED On
```

```
  digitalWrite(green, HIGH);
  digitalWrite(blue, HIGH);
}

// Green LED ON function
void ledGreen()
{
  digitalWrite(red, HIGH);
  digitalWrite(green, LOW);  // Turn on the RGB Green LED On
  digitalWrite(blue, HIGH);
}

// Blue LED ON function
void ledBlue()
{
  digitalWrite(red, HIGH);
  digitalWrite(green, HIGH);
  digitalWrite(blue, LOW);  // Turn on the RGB Blue LED On
}
```