

# Wine Room Remote Temperature Monitor Project

---

*Bill Ballard*

We have a wine storage unit in our garage, and over the last 20 years the compressor has failed during extreme heat waves twice. Other than manually looking at the interior temperature, we had no warning of impending failure that would have allowed us to move the wine to an air conditioned location and have the compressor repaired. So, I developed this project to monitor the temperature hourly, and send an email when it was above the “alarm” level, and another email if it got above a “fail” level. Now I’ve updated the instructions for a Raspberry Pi Zero W, but the original is still running off an A+, the only difference would be the need for a WiFi dongle.

## Hardware Needed

- Model Zero W Raspberry Pi
- 8 GB uSD card (4 GB is sufficient if you can find one)
- Suitable case for the Pi
- Waterproof DS18B20 Temperature Sensor (Adafruit product 381)
- Stacking header (can be 26 pin version as only a part of the header is needed)
- Perfboard and a few bits of wire unless you are really handy with a soldering iron
- Suitable Power 5V USB supply
- Empty wine bottle and a cork that fits it
- Adapters to connect HDMI and USB to real components for the setup phase

## Software Setup

Download and install the latest version of Raspian Jessie **Lite** on your micro SD card using any of the excellent instructions in the Raspberry Pi Forum. Temporarily hook the system up to a monitor, keyboard, and mouse. Boot it up and initialize Raspian with raspi-config to the proper international options (time zone, keyboard, etc.), change the default password, set the host name in advanced options to something like wine or winerom so you know which Raspberry Pi you are talking to (don’t you have more than one?). Make sure that you enable both SSH and 1-Wire in the Interface section of raspi-config.

Reboot (otherwise your keyboard may not be correct) and set up the wireless. I like simply editing:

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

And add these lines to the end.

```
network={  
    ssid="your ssid"  
    psk="your password"  
    key-mgmt=WPA-PSK
```

```
}
```

be sure to replace “your ssid” and “your password” with the actual values for your network and keep the quotes. You can also set your Country code here or with raspi-config. Save the file with Control-o, and then exit with Control-x. Now reboot with

```
$ sudo reboot
```

When you start back up, you will see the ip address assigned to the Zero W or you can use the following command to see the ip V4 address that you will need to SSH into the pi later. Write it down (192.168.1.27 or 10.0.0.27 are typical values).

```
$ iwconfig
```

Now you should run, and this can take quite some time if it is the first time you are doing it:

```
$ sudo apt-get update && sudo apt-get upgrade -y
```

Next install the mail and ssmtp applications:

```
$ sudo apt-get install ssmtp heirloom-mailx
```

And edit the ssmtp configuration file to point to your gmail account as shown below. If you don't have a gmail account, I strongly recommend getting one, and only using it for this purpose to minimize hacker issues. I haven't had much luck using other mail services for this project. You need to add the text in red further on down to the configuration file.

```
$ sudo nano /etc/ssmtp/ssmtp.conf
```

```
#
```

```
# Config file for sSMTP sendmail
```

```
#
```

```
# The person who gets all mail for userids < 1000
```

```
# Make this empty to disable rewriting.
```

```
# root=postmaster
```

```
# The place where the mail goes. The actual machine name is required
```

```
# no MX records are consulted.
```

```
# Commonly mailhosts are named mail.domain.com
```

```
# mailhub=mail
```

```
# Where will the mail seem to come from?
```

```
# rewriteDomain=
```

```
# The full hostname (default below)
```

```
# hostname = raspberrypi
```

```
# Are users allowed to set their own From: address?
```

```
# YES - Allow the user to specify their own From: address
```

```
# NO - Use the system generated From: address
```

```
# FromLineOverride=YES
```

```
# Wine Room Monitor settings – for gmail
hostname=winerom
root=your-mail@gmail.com
mailhub=smtp.gmail.com:587
FromLineOverride=YES
AuthUser=your gmail name (leave out the @ and stuff after that)
AuthPass=your gmail password (no quotes needed here)
AuthMethod=LOGIN
UseSTARTTLS=YES
UseTLS=YES
```

Now do a control-o to write the file and a control-x to exit. As we will run the monitor program with crontab, we also must change the /etc/ssmtp/revaliaes file or authentication errors will occur.

```
$sudo nano /etc/ssmtp/revaliaes
```

and add the following line.

```
root:your-email@gmail.com:smtp.gmail.com:587
```

Now reboot to get everything set up with the changes.

```
$ sudo reboot
```

## The Program

Log back in to the pi. First, create a directory called winerom and move into it:

```
$ mkdir winerom
```

```
$ cd winerom
```

Next, let's enter the sensor code in the file ds18b20.c:

```
$ nano ds18b20.c
```

and paste in the following code:

```
// be sure /boot/config.txt has this line, but raspi-config should have added it at the end
// dtoverlay=w1-gpio
```

```
// Compiling & Running
```

```
// Assuming the code is saved in a file called ds18b20.c,
// run the following to compile -
```

```
//gcc -Wall -std=gnu99 ds18b20.c -o ds18b20
```

```
//Run the code by typing ./ds18b20.
```

```
#include <stdio.h>
#include <ctype.h>
#include <stdint.h>
#include <dirent.h>
#include <string.h>
#include <fcntl.h>
```

```

#include <stdlib.h>
#include <unistd.h>
#include <time.h>

int main (void) {

// C requires that we declare ALL variable types, some we initialize

    DIR *dir;
    FILE *fp;
    struct dirent *dirent;
    char email[]="your-email@gmail.com";    // this must be your own e-mail account!
    char dev[16];    // Dev ID
    char devPath[128]; // Path to device
    char buf[256];    // Data from device
    char tmpData[6]; // Temp C * 1000 reported by device
    char path[] = "/sys/bus/w1/devices";
    ssize_t numRead;
    float alarm=60.5;
    float fail=63.0;
    float tempC=0.0;
    float tempF=0.0;
    int ix;
    char out_string[200];
    char *sys_string;
    time_t current_time;
    current_time=time(NULL);
    out_string[0]='\0';

// now lets find the 1-wire files

    dir = opendir (path);
    if (dir != NULL)
    {
        while ((dirent = readdir(dir)))

// 1-wire devices are links beginning with 28-

            if (dirent->d_type == DT_LNK &&
                strstr(dirent->d_name, "28-") != NULL) {
                strcpy(dev, dirent->d_name);
// debug line    printf("\nDevice: %s\n", dev);
            }
            (void) closedir (dir);
        }
        else
        {
// nobody is there
            perror ("Couldn't open the w1 devices directory");
            return 1;
        }

// Assemble path to OneWire device

        sprintf(devPath, "%s/%s/w1_slave", path, dev);

// Read temp
// Opening the device's file triggers new reading

```

```

int fd = open(devPath, O_RDONLY);
if(fd == -1)
{
    perror ("Couldn't open the w1 device.");
    return 1;
}

// read the data and load it into our variables

while((numRead = read(fd, buf, 256)) > 0)
{
    strncpy(tmpData, strstr(buf, "t=") + 2, 5);
    tempC = strtouf(tmpData, NULL);
    tempC = tempC/1000;
    tempF = tempC*9./5.+32.;
}
close(fd);

// Now, write to log file a formatted output

fp = fopen("/home/pi/wineroom/wine.log", "a");
fprintf(fp,"%5.1f\t%s", tempF, ctime(&current_time));
fclose(fp);

// send e-mail if over temperature. Be sure you set your own e-mail in the initialization
// note multiple line sprintf, a semi-colon is needed to end the line

if(tempF > fail) {
    sprintf(out_string, "echo \'wine room fail %5.1f F\' | "
    "mail -s \'wine room fail %5.1f F\' %s",
    tempF, tempF, email);
    sys_string = out_string;
    ix = system(sys_string);
} else if (tempF > alarm) {
    sprintf(out_string, "echo \'wine room alarm %5.1f F\' | "
    "mail -s \'wine room alarm %5.1f F\' %s",
    tempF, tempF, email);
    sys_string = out_string;
    ix = system(sys_string);
}
return ix;
}

```

Of course, I hope you just downloaded that and pasted it into a ds18b20.c file. To compile:

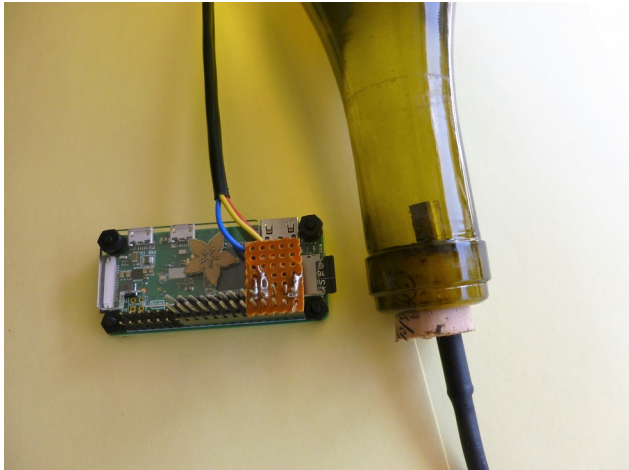
```
$ gcc ds18b20.c -o ds18b20 -Wall -std=gnu99
```

## Hardware Setup

Take the wine bottle cork and cut it in half with a hacksaw. Carefully clamp it and use a 1/4" drill bit to drill a hole through the center of the cork. Next push the ds18b20 sensor stainless steel through the cork so that it is protruding through the cork – this is why we cut it in half. Wetting the sensor may make insertion easier. Now fill the wine bottle nearly full of water after rinsing it out with hot water. Wet the cork and push it carefully into the bottle. Then put the bottle on its

side on a piece of newspaper to check for leaks. You may want to add a bit of silicone to seal around the sensor if you do have a leak.

Next we need to set up the sensor. Mount a small bit of perf board on a stacking header and connect the ds18b20 red wire to Raspberry Pi Pin1 (3.3V), the ds18b20 white wire to Raspberry Pi Pin 4, and a 4.7k resistor (provided with the sensor) between these two pins. Finally, the ds18b20 black wire connects to the Pi's Pin 5, a ground, or any other ground pin you choose. This requires a bit of patience and soldering. It is possible to do this without the perf board directly on the header if you like.



Now plug it onto the Raspberry Pi (POWERED OFF!) being careful to get the location of pin 1 right, boot the pi up, log in and test it with:

```
$ cd winerroom
$ sudo ./ds18b20 &
```

Look at the file in your /home/pi/winerroom directory called wine.log, there should be one entry. Also, you are probably way over the temperature range and should get an e-mail showing an alarm or alert. This will test whether you've got the e-mail settings correct. If you don't get an email in a few minutes, take a look at /etc/ssmtp/ssmtp.conf again to see if there are any errors.

Now we create a short script to rename the log file and add the year and month to the log file name at the end of every month. Of course, since we run it on the first, we need last month's date.

```
$ nano logupdate.sh
```

```
#!/bin/bash
STR=$(date -d "last month" +"%y-%m")
mv /home/pi/winerroom/wine.log /home/pi/winerroom/$STR-wine.log
mail -a /home/pi/winerroom/$STR-wine.log -s "Monthly Wine Log" your-email@gmail.com
```

Control-o, control-x to write this out and make it executable with:

```
$ chmod +x logupdate.sh
```

Now, we want the monitor code to run hourly, and start a new wine log every month, and we do this with crontab. Edit the crontab with the command below and add the two lines shown to the bottom of the file. The first line runs the code at 5 minutes after the hour, every hour of every day (lots of stuff seems to run on the hour), and the log file renaming happens at 4 minutes after midnight on the first of the month.

```
$ sudo crontab -e
```

```
5 * * * * /home/pi/wineroom/ds18b20
```

```
4 0 1 * * /home/pi/wineroom/logupdate.sh
```

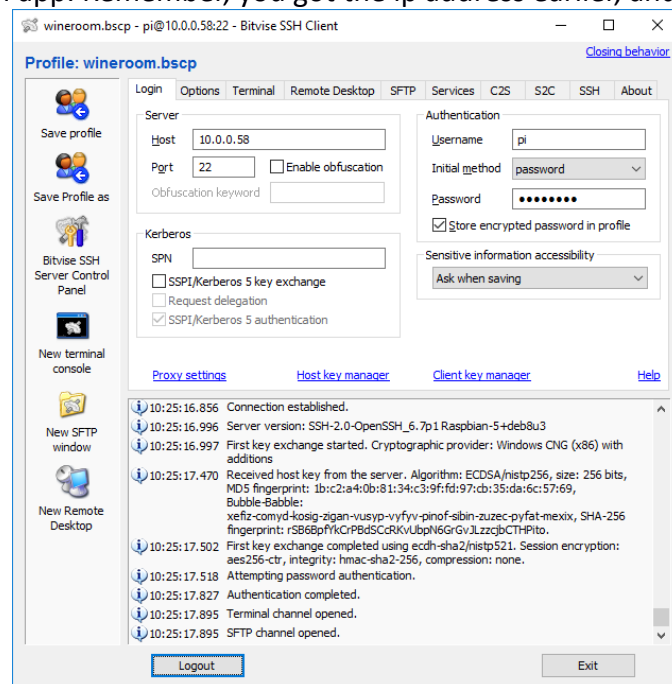
Exit the file with control-o and control-x and do the following to get the new crontab entries working.

```
$ sudo service cron start
```

```
$ sudo update-rc.d cron defaults
```

To install in the wine room after testing everything out, I used some Velcro on the bottom of the Raspberry Pi Zero W case to attach it to the cooler unit. I removed a bit of the sealant around the cooler and fished a miniUSB power cable through so I could power externally though you may have an accessory plug available on your compressor inside the wine room. Then I replaced the sealant (butyl rubber) to close the system back up. I powered it up, and let it run.

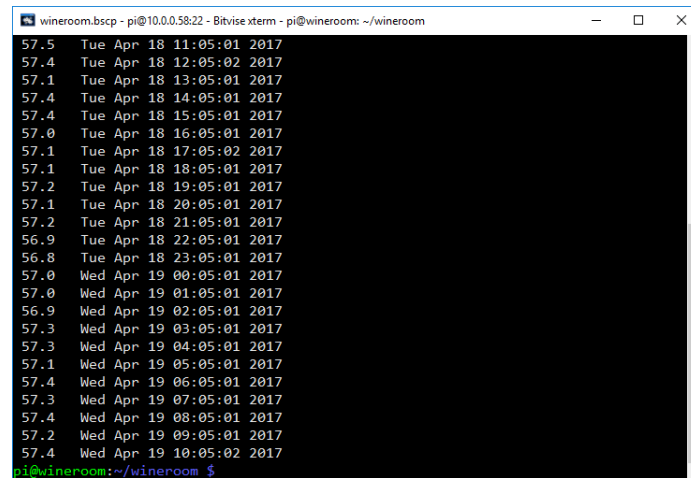
To check in with the temperature monitor, I use BitVise (a great free replacement for Putty from the same folks) from a windows computer or ssh from another Linux computer. I also use an iPad with the WebSSH app. Remember, you got the ip address earlier, and will need it now.



Or in Linux from a terminal window do (use your own IP address for the pi)

```
$ssh pi@10.0.0.58
```

And log in. A `tail -n 24 ~/winerom/wine.log` will get you the last day's temperatures.

A screenshot of a terminal window titled "winerom.bscp - pi@10.0.0.58:22 - Bitvise xterm - pi@winerom: ~/winerom". The terminal displays the output of the command `tail -n 24 ~/winerom/wine.log`. The output consists of 24 lines of log entries, each showing a temperature value, a day of the week, a date, and a time. The temperatures are mostly in the 56-57 degree Fahrenheit range. The log entries span from Tuesday, April 18, 2017, to Wednesday, April 19, 2017. The terminal prompt at the bottom is `pi@winerom:~/winerom $`.

```
57.5 Tue Apr 18 11:05:01 2017
57.4 Tue Apr 18 12:05:02 2017
57.1 Tue Apr 18 13:05:01 2017
57.4 Tue Apr 18 14:05:01 2017
57.4 Tue Apr 18 15:05:01 2017
57.0 Tue Apr 18 16:05:01 2017
57.1 Tue Apr 18 17:05:02 2017
57.1 Tue Apr 18 18:05:01 2017
57.2 Tue Apr 18 19:05:01 2017
57.1 Tue Apr 18 20:05:01 2017
57.2 Tue Apr 18 21:05:01 2017
56.9 Tue Apr 18 22:05:01 2017
56.8 Tue Apr 18 23:05:01 2017
57.0 Wed Apr 19 00:05:01 2017
57.0 Wed Apr 19 01:05:01 2017
56.9 Wed Apr 19 02:05:01 2017
57.3 Wed Apr 19 03:05:01 2017
57.3 Wed Apr 19 04:05:01 2017
57.1 Wed Apr 19 05:05:01 2017
57.4 Wed Apr 19 06:05:01 2017
57.3 Wed Apr 19 07:05:01 2017
57.4 Wed Apr 19 08:05:01 2017
57.2 Wed Apr 19 09:05:01 2017
57.4 Wed Apr 19 10:05:02 2017
pi@winerom:~/winerom $
```

The air temperature in a wine room cycles pretty widely, but the temperature of the wine is held quite stable so you should see only minor changes in the temperature. If needed, you can adjust the alarm and fail temperature levels in the program, or switch from Fahrenheit to Celsius temperatures. Enjoy, and I hope your compressor never fails in a 115°F (46 C) heat wave!