RV Temperature Monitor Project

Bill Ballard

My sister and her husband have an RV and a dog they dearly love. They take him on all their trips. Occasionally they would like to leave him in the RV if they are hiking somewhere dogs shouldn't go (snakes, excessive heat, …) so they asked me to make a system to alert them if the RV air conditioning failed.

This project uses a Raspberry Pi Zero W, and the BerryLan distribution so that the headless configuration can learn the WiFi credentials for whichever RV park they are staying in.

Hardware Needed

- Model Zero W Raspberry Pi
- 4 GB uSD card (larger card is just fine)
- Suitable case for the Pi
- DHT11 Temperature Sensor and resistor (Adafruit product 386)
- Header for the GPIO
- Suitable Power 5V USB supply

Software Setup

Download and install the latest version of BerryLan (based on Stretch lite**)** on your micro SD card using any of the excellent instructions in the Raspberry Pi Forum. Download the BerryLan app from the Apple or Android store (a freebie). Boot up the ZeroW and give it a few minutes to set things up. Then start the app and find the Raspberry Pi. Now you can enter your home SSID and password to your WiFi network.

Next remote in with Putty or Bitvise (user:pi, Password: raspberry, Hostname:raspberry.local), 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 Rvtemp so you know which Raspberry Pi you are talking to (don't you have more than one?).

Reboot (otherwise your keyboard may not be correct). Then you can sign in with the hostname Rvtemp.local and the new password using Putty or Bitvise.

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, ssmtp, and wiringpi applications:

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

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 if 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=wineroom
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/revaliases file or authentication errors will occur.

```
$sudo nano /etc/ssmtp/revaliases
```

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.

Next, let’s enter the sensor code in the file dht11.c:

```
$ nano dht11.c
```

and paste in the following code:

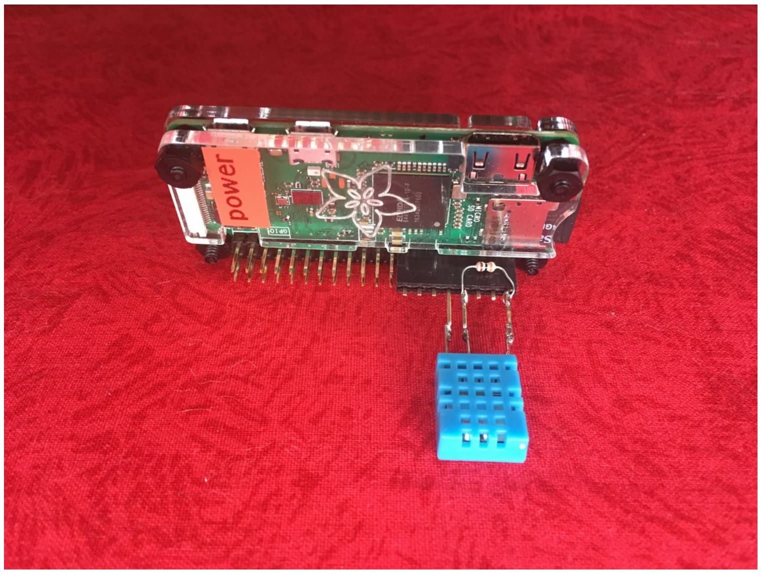
```
// dht11.c
// compile line
// gcc -Wall -std=gnu99 temp11.c -o temp11 -l wiringPi

#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>
#define MAX_TIME 85
#define ATTEMPTS 5
#define DHT11PIN 7
int dht11_val[5]={0,0,0,0,0};

int dht11_read_val()
{
    float humid=0.0;
    float tempC=0.0;
    float tempF=0.0;
    uint8_t lststate=HIGH;
    uint8_t counter=0;
    uint8_t j=0,i;
    int addr, nemail;
    char out_string[200];
    time_t current_time;
    current_time=time(NULL);

// here are the text equivalents for various phone companies if you prefer
// text messages to plain email
//
// AT&T: [number]@txt.att.net
// Sprint: [number]@messaging.sprintpcs.com or [number]@pm.sprint.com
// T-Mobile: [number]@tmomail.net
// Verizon: [number]@vtext.com
// Boost Mobile: [number]@myboostmobile.com
// Cricket: [number]@sms.mycricket.com
// Metro PCS: [number]@mymetropcs.com
// Tracfone: [number]@mmst5.tracfone.com
// U.S. Cellular: [number]@email.uscc.net
// Virgin Mobile: [number]@vmobl.com

// customize the next four lines for your needs, temperatures in Farenheit
    float alarmhi=80.0;
    float alarmlo=40.0;
    const char *email[2]={"phonenumber@vtext.com","youremail@gmail.com"};
    const char *loc="Your RV";

    nemail = sizeof(email)/sizeof(email[0]);
    for(i=0;i<5;i++)
         dht11_val[i]=0;

    pinMode(DHT11PIN,OUTPUT);
    digitalWrite(DHT11PIN,LOW);
    delay(18);
    digitalWrite(DHT11PIN,HIGH);
    delayMicroseconds(40);
    pinMode(DHT11PIN,INPUT);

    for(i=0;i<MAX_TIME;i++)
    {
        counter=0;
        while(digitalRead(DHT11PIN)==lststate){
            counter++;
            delayMicroseconds(1);
            if(counter==255)
                break;
        }
        lststate=digitalRead(DHT11PIN);
        if(counter==255)
             break;
// top 3 transistions are ignored
        if((i>=4)&&(i%2==0)){
            dht11_val[j/8]<<=1;
            if(counter>16)
                dht11_val[j/8]|=1;
            j++;
        }
    }

// verify checksum and print the verified data
// dht11 has only one digit precision
    if((j>=40)&&(dht11_val[4]==((dht11_val[0]+dht11_val[1]+dht11_val[2]+dht11_val[3])&
0xFF)))
    {
        humid=1.*dht11_val[0];
        tempC=1.*dht11_val[2];
        tempF=(9.*tempC/5.)+32.;

// uncomment the next line to test email system
//       tempF=100,;

        printf("%5.1f C\t%5.1f F\t%5.1f\t%s",tempC,tempF,humid,ctime(&current_time));

// send e-mail if over or under temperature limits
        if(tempF > alarmhi) {
            for (addr=0; addr<nemail; addr++) {
                sprintf(out_string,
                "echo '%s high temp %5.1f F alarm at %s' | "
                "heirloom-mailx -s '%s high temp %5.1fF alarm' %s",
                loc, tempF, ctime(&current_time), loc, tempF, email[addr]);
//              printf("%s\n",out_string);
                system(out_string);
            }
        } else if (tempF < alarmlo) {
            for (addr=0; addr<nemail; addr++) {
                sprintf(out_string,
                "echo '%s low temp %5.1f F alarm at %s' | "
                "heirloom-mailx -s '%s low temp %5.1fF alarm' %s",
                loc, tempF, ctime(&current_time), loc, tempF, email[addr]);
//              printf("%s\n",out_string);
                system(out_string);
            }
        }
        return 1;
    }
    else
        return 0;
}

int main(void)
{
    int attempts=ATTEMPTS;
    if(wiringPiSetup()==-1)
        exit(1);
    while(attempts)
    {
        int success = dht11_read_val();
        if (success) {
            break;
        }
        attempts--;
        delay(500);
    }
    return 0;
}
```

Of course, I hope you just downloaded that and pasted it into a dht11.c file. To compile:

```
$ gcc dht11.c –o dht11 –Wall -std=gnu99 -l wiringPi
```

Hardware Setup

Solder a GPIO header into the 40-pin holes on the ZeroW. If you are brave, you can skip this and solder the temperature sensor directly to the board, but I think you are better off using a header. The DHT11 temperature sensor has 4 pins, but only 3 of them are active. With the openings on the sensor facing you, pin 3 has no utility and I cut it off. Next, we will connect to pins 1, 7 and 9 of the GPIO so bend pin 1 and pin 2 of the DHT11 to align with these pins. Next solder the 10k resistor between pins 1 and 7 of the header. Then solder the DHT11 to the three active header pins. I cut off the unused header pins to make access easier.



Now plug it onto the Raspberry Pi (POWERED OFF!) being careful to get the location of pin 1 right (it is closest to the uSD card slot). Boot the pi up and reconnect with Putty or Bitvise. Test the file by

```
$./temp11
```

and see if you get temperature and humitidy information printed out. To test the email setup, simplest is to edit the temp11.c file and uncomment the temp=100.; line, save the file and recompile. This will force an email when you run the code so you can be sure it is all working. Then edit the temp11.c file and comment out the temp=100.; line, recompile and you are good to go.

We will run the temperature code every 10 minutes with cron, so type

```
$crontab -e
```

and add the following line at the end of the file

```
*/10 * * * * /home/pi/temp11 >/home/pi/temp.log 2>&1
```

Save with control-O and exit with control-X. This crontab file will execute temp11 every 10 minutes of every day and hour, writing the current temperature, humidity and time to the log file. You can run this system off a phone battery charger with a 2A output if you are concerned over power failure into the RV itself.

Usage

When you arrive at the remote location and check in, you should receive the ssid and password to the WiFi on the site. Power up the Raspberry Pi and give it a minute or so to boot up. Then use the BerryLan app you downloaded from the Apple store. The Raspberry Pi will discover that there is no known WiFi available and start a BlueTooth connection to the phone. You can select the proper computer and then enter the ssid and password. The Pi will then connect to the WiFi and you are all set.