Temperature monitor

Arduino - NAS

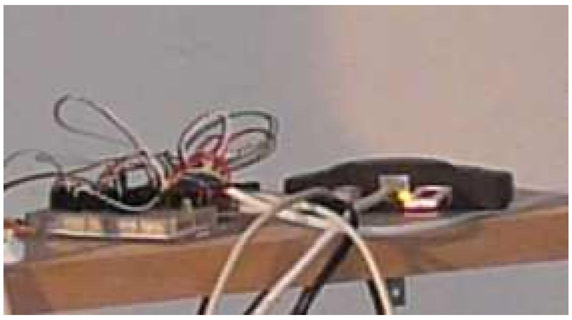
Emmanuel Pottier - 2011-11-02

Goal

The goal of this project is to create a remote temperature measurement system, with data logger capability & possibility to read data over the internet.

Description

I built the system based on two items I already had: an Arduino and a USB-NAS server (model 652WLXKJ), normally used to provide network access to a USB drive.



General view. See size of the usb stick to have an idea of the complete size

Principle is:

1. The arduino read a temperature sensor
2. It sends the value to the USB port of the NAS
3. The NAS copy this value in a file, on a USB stick
4. The file is than available on the NAS, as a network share or via a direct ssh/scp connection.

Advantages to use a NAS serveur are:

- Native acces to the network
- 2 usb port (one will be for the arduino, the other for a usb stick)
- an alternative firmware is available (Snake OS), allowing full access to the embedded Linux
- the NAS is able to power the arduino, what remove the need of alternative power
- power consumption is very low (a few watts)

So, for only 30€, the NAS is a very good data logger, ethernet shield & power. It's also very evolutive for future applications (200Mhz processor, 256Mb of memory, possible to plug external drives of many Gb).

The temperature measurement itself will be done by the arduino, with TMP36 sensors, plugged on the analog entries.

Parts needed / Prerequisites

You need the following :

- A USB NAS server, model 652WLXKJ, available at "dealextreme" (http://www.dealextreme.com/p/standalone-bittorrent-bt-client-upnp-usb-nas-ftp-samba-printer-sharing-network-lan-server-20383)

  (Note: it seem that some USB NAS uses a different chip, so are not compatible with Snake OS)
- An Arduino board (I have an Arduino Uno, but any clone should be ok)
- A tmp36 sensor
- A USB stick (I used an old 128Mb one)

Quick estimation is 30 € for the NAS, same for the arduino + 2 sensors and a USB stick so total around 60 €.

I assume that the reader is familar with the Arduino, knows how to connect via ssh on a machine, and flash a firmware.

Programmation

Arduino

The Arduino sketch is pretty basic, as it only need to read the sensor, then send the values on the serial port.

The schema is explained on the "ladyada" website: http://www.ladyada.net/learn/sensors/tmp36.html

The TMP36 sensor have the advantage to be easy to use for temperatures inside an house, including negative values (Celsius). The formula to get the temperature is the following: T°C = (V-500)/10

Where V is the voltage, on sensor pin, in mV.

It's interesting to see that 1v as output represent 50°C, what is pretty hot inside an house. We can assume we will never measure more than this. Then, we can set the analog ref of the Arduino to "INTERNAL", so "1.1" volts, and get maximum accuracy.

Then the formula is: T°C = (analogRead(sensorPin)*1100/1024-500)/10

Important: due to float management on the Arduino, it's better to do the multiplications first and divisions after, and in general, to limit the number of operations. Also it's always good to add a ".0" to integers, so to be sure Arduino operate as floats, and does not convert to integers.

The formula become: T°C = (analogRead(sensorPin)*1.074-500.0)/10.0

In practice, the internal reference is not exactly 1.1v. In my case, after some measurement, the correct formula is: T°C = (analogRef(sensorPin)*1.06157-500.0)/10.0
Remark: you might need to adjust the formula to get the result in °Farenheit.

Here is the final script:

```
void setup(){
   // Declare analog reference
   analogReference(INTERNAL);

   // Declare serial connection
   Serial.begin(9600);
}

float ReadTemp(int analogPin){
   // Read temperature on a specified pin
   float sensorTemp = (analogRead(analogPin)*1.06157-500)/10.0;
   return sensorTemp;
}

void loop(){
   float TmpInt = ReadTemp(0); // The sensor inside the house is
on port 0
   float TmpExt = ReadTemp(1); // The sensor outside the house is
on port 1

   // Write values on the serial port
   Serial.print(TmpInt);
   Serial.print(";");
   Serial.print(TmpExt);

   // Wait before next measurement
   delay(10000);
}
```

USB-NAS

Firmware update: by default, the NAS is shipped with a firmware that does not allow customisation. So it's needed to flash it and install "Snake OS", available on google code (http://code.google.com/p/snake-os/). The upgrade process is explained on the project site, and should not be a problem.

It is then needed to make sure the Arduino is able to talk with the NAS. For the Arduino that use an ftdi USB chip (most, except Arduino Uno), it should be automatic. For Arduino Uno, it's another USB chip, what require a specific module "cdc-acm" (see below how to load the module).

Once the Arduino is plugged in the NAS, you need to connect with ssh on the NAS, then type "dmesg", to see if the arduino is well connected. It should show something similar to this:

```
usb 2-2: new full speed USB device using str8100-ohci and address
2

usb 2-2: configuration #1 chosen from 1 choice

cdc_acm 2-2:1.0: ttyACM0: USB ACM device
```

Here the Arduino is identified as ttyACM0 (value might be different for Arduino clones). It will be reachable via /dev/ttyACM0.

If the last line is missing, it means the arduino is not recognized.

Once recognized, it's possible to read what the arduino send to the USB port by typing: cat /dev/ttyACM0.

Then you should see the temperatures, at each second, something like: 21.45;16.39

As we want to send the result to a file, we will create a redirect, inside a "nohup", so the redirect is still active when we disconnect ssh.

```
nohup cat /dev/ttyACM0 >> /usb/sda1/data.log &
```

You can type "top" to see that the "cat" command is still executed in background.

We can now read the "data.log" file on the usb stick to get the data. This is doable via an ssh connection, then "cat /usb/sda1/data.log" or you can configure in snake-os a file share on the USB stick. Then you can access the data directly with your file explorer.

It's also interesting to insert in the file a timestamp. This is done by going to the NAS, then edit /etc/cron.d/root, and add :

```
0 * * * * date +%F_%T >> /usb/sda1/data.log
```

Note: it is needed to save the file, then save NAS configuration (via the NAS web interface), then restart the NAS so the task is scheduled.

For the Arduino Uno

The Arduino Uno uses a special usb chip, and so require a specific module in the NAS. This module, cdc-acm.ko is available on download (I generated it with the snake-os sdk: http://pottier.emmanuel.free.fr/cdc-acm.ko).

You need to put it on the usb stick, then launch: insmod /usb/sda1/cdc-acm.ko (if everything is fine, this should return nothing).

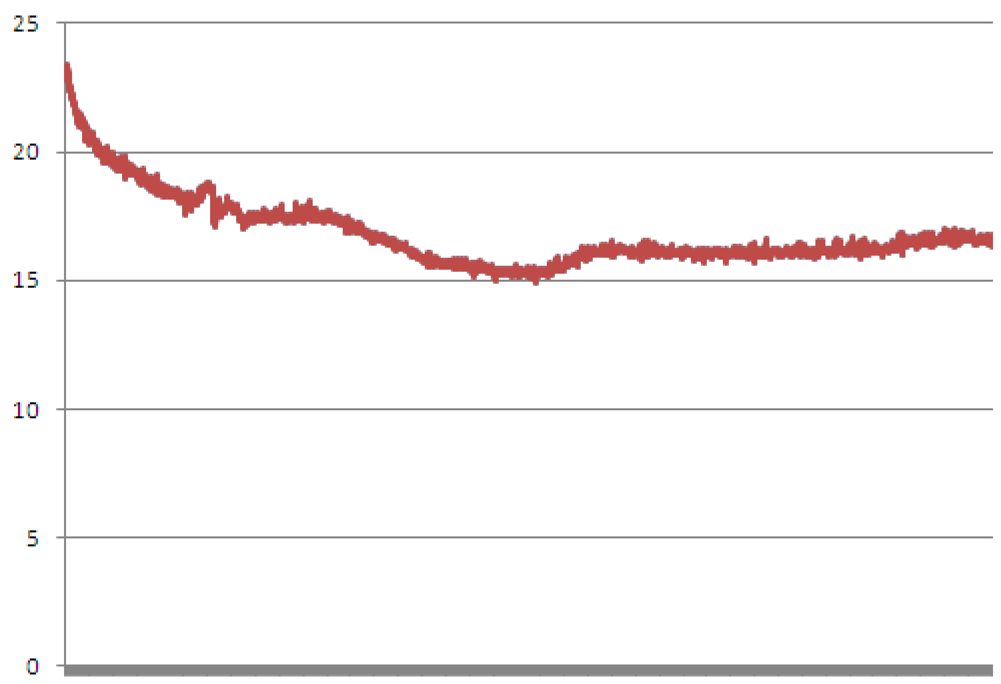
Where to start?

If you are a bit lost, here is a summary how to proceed.

1. Start with the Arduino
   a. Wire the TMP36 sensor
   b. Write the sketch, check on the serial monitor that you receive the values
2. On the NAS
   a. Flasher snake-os firmaware
   b. Connect with ssh, and check everything is working fine
3. Connect Arduino to the NAS
   a. Check with dmesg that the Arduino is well connected
   b. Check with cat /dev/ttyACM0 that you can read the values
   c. Setup the "nohup", so the temperatures are automatically copied to the usb stick
   d. Setup the "cron" task, to get the timestamps

Going further

This is my first "really usable" system Arduino based. I use it to get the temperature in an house that I do not occupy the entire year.



Resultst over 3 days, begining of November
(post-treatment in excel)

There is a lot of possible evolutions, for instance:

- use the web serveur integrated in the NAS to display temperature, graphs, on a web page
- trigger remote actions on the Arduino, for instance to drive heating system. This would need to send data from the NAS to the Arduino. This should be possible, but we would have to test this.
- automatically send email from the NAS (why not a "frozing alert"), this is theorically possible, but it seem the NAS does not have a send mail option
- it's currently needed to restart the "nohup" task at each reboot, it would be nice to make this automated at startup

Links

- TMP36 on Arduino: http://www.ladyada.net/learn/sensors/tmp36.html
- Snake-os firmware: http://code.google.com/p/snake-os/
- NAS at dealextreme (clones should exist): http://www.dealextreme.com/p/standalone-bittorrent-bt-client-upnp-usb-nas-ftp-samba-printer-sharing-network-lan-server-20383
- Arduino board, in any Arduino reseller, I used this one to start with: http://www.oomlout.co.uk/prototyping-bundle-for-arduino-ardp-p-186.html
- cdc-acm.ko file for Arduino Uno : http://pottier.emmanuel.free.fr/cdc-acm.ko

Thanks to...

Ricardo Gomes da Silva, that explained me how to complie cdc-acm
Douglas Gazineu, who created the snake-os firmware and the SDK
And of course, to the entire Arduion team, that created this marvelous tool.