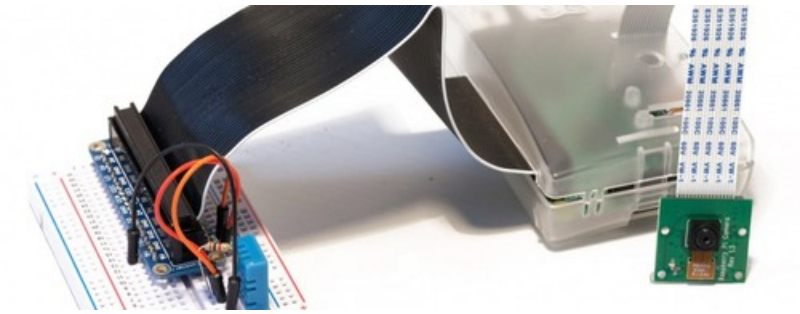
adafruit learning system

Monitor Your Home With the Raspberry Pi B+

Created by Marc-Olivier Schwartz



Last updated on 2018-08-22 03:46:29 PM UTC

Guide Contents

Introduction

There are lots of devices on the market that allow you to monitor your home from a central interface. In this article, we are going to make our own DIY version of such devices. To do so, we will use the latest Raspberry Pi board, the B+ model, and the official Raspberry Pi camera module (https://adafru.it/euT). We will also perform some measurements from a temperature and humidity sensor.

At the end of the article, you will be able to build an interface to access the camera and the sensor recordings from a single page. We will also see how to acess this interface from anywhere in the world. Let's dive in!

Hardware & Software Requirements

The first thing that you will need for this project is a Raspberry Pi B+. I used a B+ model as it is the latest version available to date. It also has nice features (like 4 USB ports), but of course, you can also use an older version.

You will need the official Raspberry Pi camera module to capture pictures. You will use a DHT11 (or DHT22) sensor to measure the temperature and humidity in your home.

Since we will access the Rapsberry Pi remotely, you will need a simple USB WiFi dongle.

You also need to secure an Adafruit cobbler kit, a breadboard, and some jumper wires. You will need these to make the connections between the Raspberry Pi, the camera and the sensor.

Below is the list of the required components for this project:

- Raspberry Pi B+ (along with a microSD card, a microUSB cable, and an HDMI cable)
- Raspberry Pi camera module
- DHT11 sensor with 4.7k Ohm resistor
- USB WiFi dongle
- Adafruit Cobbler Kit
- Jumper wires
- Breadboard

Check if you already have a Linux Distribution installed on your Raspberry Pi. This is to ensure that you have a completely functional Pi. I used Raspbian for this project.

If this is not done yet, you can find an excellent guide on this address:

http://www.raspberrypi.org/help/quick-start-guide/ (https://adafru.it/euU)

Connect the Raspberry Pi to your local WiFi network and install a driver for the BCM2835 chip to read the data from the DHT11 sensor.

You can download & install these drivers by following the instructions on this page:

http://www.raspberry-projects.com/pi/programming-in-c/c-libraries/bcm2835-by-mike-mccauley (https://adafru.it/euV)

The whole project is based on Node.js. It will act as a server from which we can access all the functions of our Raspberry Pi.

If it is not done yet, you will have to install Node.js on your Pi. Be wary. You just can't use apt-get to install the node package module since you might be installing an older version. To install the latest version of Node.js, follow this guide:

http://revryl.com/2014/01/04/nodejs-raspberry-pi/ (https://adafru.it/euW)

You will also need to install drivers for the BCM2835 chip. You can download & install these drivers by visiting this page:

http://www.airspayce.com/mikem/bcm2835/ (https://adafru.it/euX)

After this, download the files for this project on GitHub:

https://github.com/openhomeautomation/rpi-web-control (https://adafru.it/euY)

To access the Raspberry Pi on your local WiFi network via rapsberrypi.local, we need to install some packages. We do this so that we wouldn't need to access the Pi via its IP address anymore.

For the rest of the article, you can either log into your Raspberry Pi via SSH or go directly to your Pi and type the following commands:

sudo apt install avahi-daemon netatalk

Hardware Configuration

If you have followed the instructions above, it would be easier to proceed with the assembly of your Pi. Let's now add the other components.

First, we will connect the camera. Follow the instructions on the Raspberry Pi website for an easy install

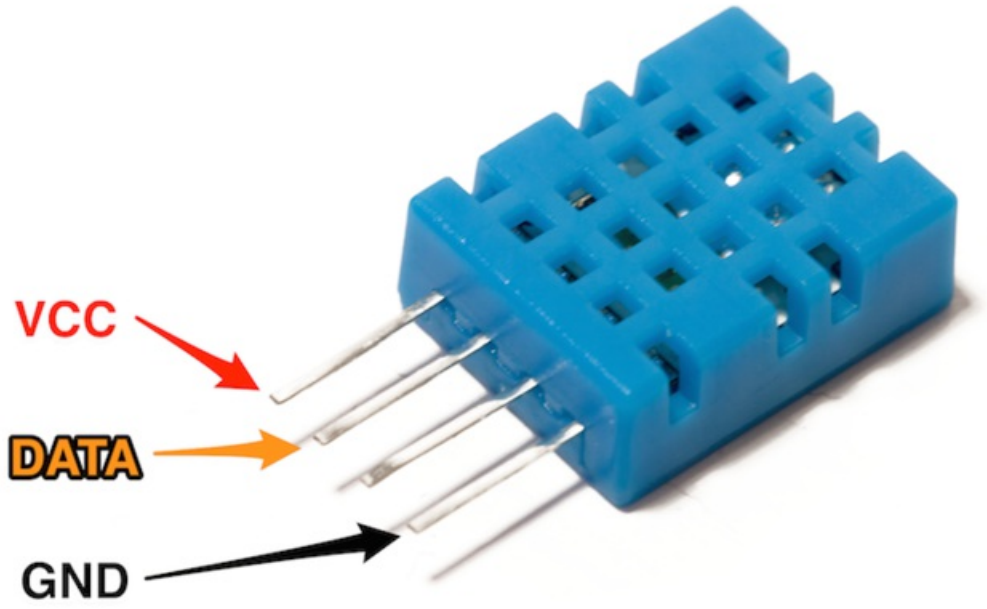
Go to the official instructions (https://adafru.it/euZ)

Below is a picture of the camera I used. Next to it is my Raspberry Pi:

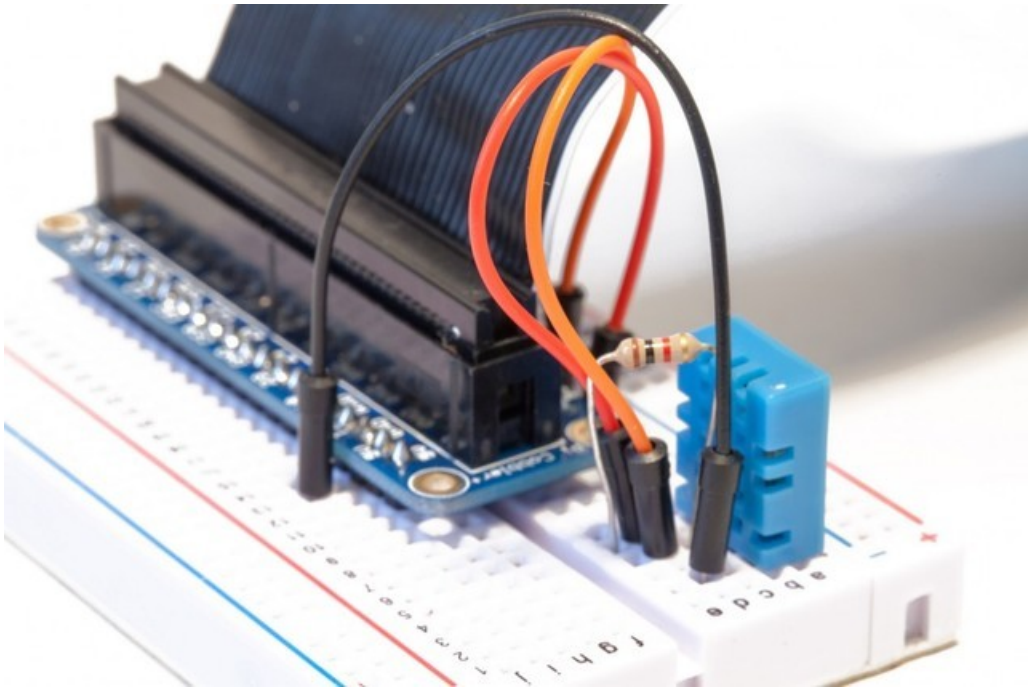
Note that I have used a plastic case for my Raspberry Pi. It is not required for this project though.

Moving forward, we will connect the DHT11 sensor to the Raspberry Pi via the Cobbler kit. After you have assembled your kit, connect it to the Raspberry Pi GPIO connector. Connect the other side of the cable to the breadboard via the PCB adapter.

For the DHT11 sensor, look first for the pinout of the sensor:



Connect the VCC pin to the Raspberry Pi 3.3V pin, GND to GND, and DATA to pin number 4 on the GPIO connector. Finally, connect the 4.7K Ohm resistor between the VCC and DATA pins. The picture below shows the final result for the sensor:



Testing the Sensor & the Camera

We are now going to test the sensor first, and then the camera.

Because we want to build an application based on Node.JS, we will use the Node to interface with the DHT11 sensor. To do so, we will use a specialized Node module that already exists.

You will find everything under a folder called sensors_test inside the code of the project. Below is the complete code for this part:

```
var sensorLib = require('node-dht-sensor');

var dht_sensor = {
    initialize: function () {
        return sensorLib.initialize(11, 4);
    },
    read: function () {
        var readout = sensorLib.read();
        console.log('Temperature: ' + readout.temperature.toFixed(2) + 'C, ' +
            'humidity: ' + readout.humidity.toFixed(2) + '%');
        setTimeout(function () {
            dht_sensor.read();
        }, 2000);
    }
};

if (dht_sensor.initialize()) {
    dht_sensor.read();
} else {
    console.warn('Failed to initialize sensor');
}
```

The code starts by importing the required module for the DHT sensor. Then, every 2000 ms, we read data from the sensor, and display it inside the terminal using console.log(). The code for this part is available in the GitHub repository of the project:

https://github.com/openhomeautomation/rpi-web-control (https://adafru.it/euY)

It's now time to test the code. After downloading the code from GitHub, go to the sensors_test folder, and type:

sudo npm install node-dht-sensor

This might take a while, so be patient. After it loads, type:

sudo node sensors_test.js

You should see the values of the temperature and humidity printed in the terminal:

Temperature: 21.00C, humidity: 35.00%

The camera is much easier to test. Simply go to a terminal window and type:

raspistill -o cam.jpg

You can then go to the folder where you executed this command. You should be able to see that a picture was created as cam.jpg.

Monitoring Your Home via WiFi

We are now going to write an application based on Node.js to remotely track the measurements from the sensor and the camera. There are basically three main parts in the code: (1) the server code in Javascript, (2) the HTML page which will contain the interface, and (3) a client-side Javascript file which will link the two.

Let's look at the server-side Javascript code first. It starts by including the required Node modules: (1) the node-dht-sensor module to handle the DHT sensor as before and (2) Express to handle HTTP communications like a web server. Below is the code:

```
var sensorLib = require('node-dht-sensor');
var express = require('express');
var app = express();
```

We also include the views and public directory. The views directory is where we will store the interface, while the public directory is where we will put both the Javascript code and the recorded pictures:

```
app.set('views', __dirname + '/views')
app.set('view engine', 'jade')
app.use(express.static(__dirname + '/public'))
```

We will create a route for the interface, which will allow us to access the our project:

```
app.get('/interface', function(req, res){
    res.render('interface');
});
```

We will include the Raspberry Pi version of the aREST API (http://arest.io/ (https://adafru.it/ev0)) for us to control the Raspberry Pi via HTTP:

```
var piREST = require('pi-arest')(app);
```

We also give an ID and a name to your Pi:

```
piREST.set_id('1');
piREST.set_name('my_RPi');
```

Finally, we call the app.listen() function to start our application:

```
var server = app.listen(3000, function() {
    console.log('Listening on port %d', server.address().port);
});
```

The interface itself is written in Jade (http://jade-lang.com/ (https://adafru.it/ev1)). This will give us an HTML file as a result. The file is stored in the view directory inside the application. We add some title, some containers for the sensor measurements, and a field for the picture which will be recorded by the camera:

```
doctype
html
  head
    title Raspberry Pi Interface
    script(src="/js/jquery-2.0.3.min.js")
    script(src="/js/interface.js")
    link(rel='stylesheet', href='/css/style.css')
  body
    header
      div.mainContainer
        h1 Raspberry Pi Interface
        h2 Sensor Readings
        p Temperature:
          span#temperature 0
          span  C
        p Humidity:
          span#humidity 0
          span  %

        h2 Camera
        img#camera(src='')
```

As you can see on the code, we will include a Javascript file and a jQuery inside the Jade template. The script file will call the Raspberry Pi aREST API every 2000 ms to refresh the measurements of the DHT11 sensor:

```
setInterval(function() {

  // Update temperature
  $.get("/temperature", function(data) {
    $("#temperature").html(data.temperature);
  });

  // Update humidity
  $.get("/humidity", function(data) {
    $("#humidity").html(data.humidity);
  });

}, 2000);
```

And every 10 seconds, the camera will take a picture:

```
setInterval(function() {

  // Take picture
  $.get("/camera/snapshot");

}, 10000);
```

This picture inside the interface will refresh every second:

```
setInterval(function() {

  // Reload picture
  d = new Date();
  $("#camera").attr("src","pictures/image.jpg?" + d.getTime());

}, 1000);
```

It is now time to test the interface. Go to the pi_node folder and type:

sudo npm install node-dht-sensor express pi-arest

Wait a bit; it can take a while.

Then, type:

sudo node pi_node.js

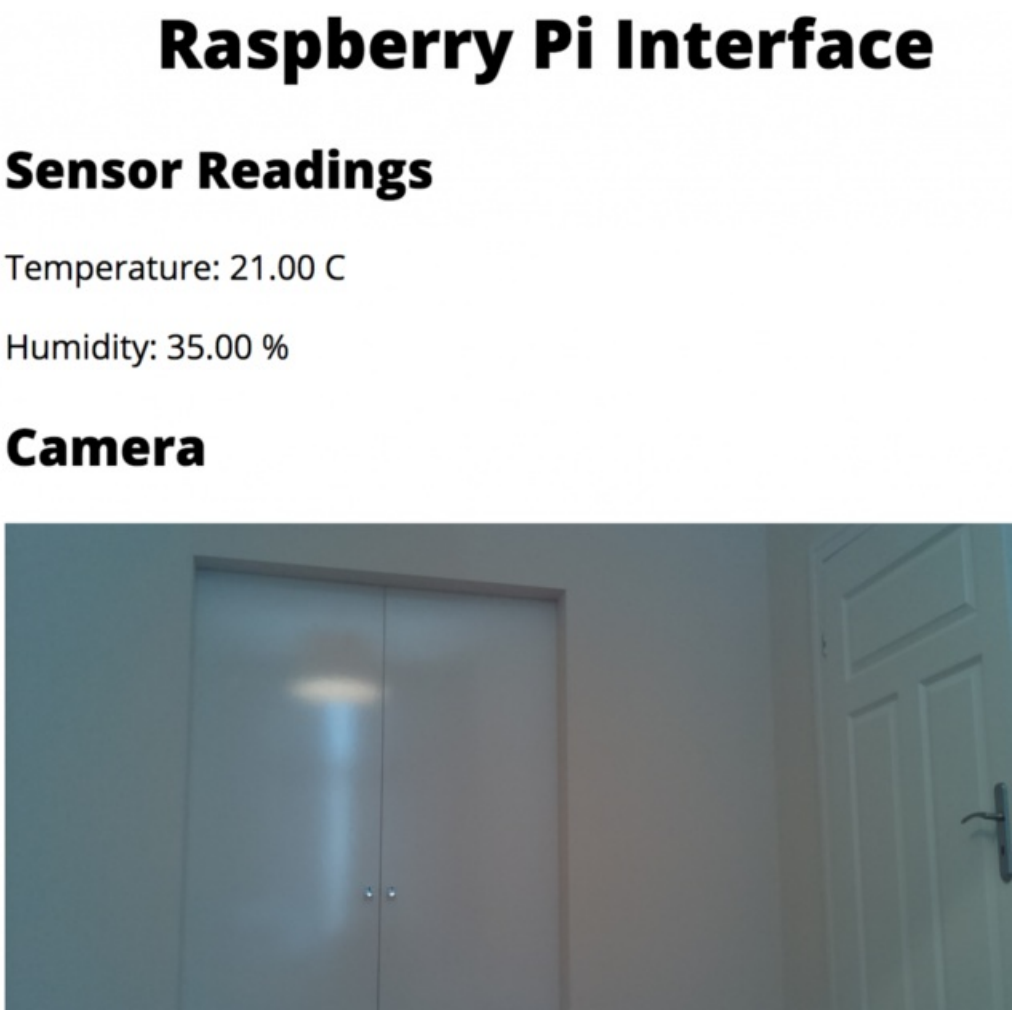
Finally, go to your favorite web browser and type:

http://raspberrypi.local:3000/interface

On your Pi, you can just type:

http://localhost:3000/interface

You should see the interface of the project displaying the following:



Be patient in waiting for the measurements and the picture to appear on the display. Remember, there is a 2-second delay for the sensor measurements and a 10-second delay for the picture.

Also, note that the Node.js module for the camera module (https://adafru.it/euT) is not perfect. In my system, I had a delay between the moment the picture was taken and the moment it appeared on the display.

Congratulations, you can now remotely access measurements from your Raspberry Pi and the camera!

Access Your Pi From Anywhere

For now, we saw how to access this interface from your computer or your phone, from anywhere ... in your home.

Which is already nice, but it would be even better if you knew how to access this project from *anywhere* in the world. Wouldn't it be cool to just access your home automation projects from wherever you are in the world, to check for example the picture recorded by the camere module? This is actually quite simple, and I will show you how.

We will use a simple tool called Ngrok to do so. This tool will basically make a tunnel between your Raspberry Pi & a remote server, so you can access your interface from anywhere. The first step is to download Ngrok:

https://ngrok.com/ (https://adafru.it/ev2)

Then, put the files in a folder, and access this folder via a terminal on your Raspberry Pi. Then, type:

./ngrok 3000

This will open the connection between the Raspberry Pi and the Ngrok server, as shown in the confirmation messages:

```
ngrok

Tunnel Status                 online
Version                       1.7/1.6
Forwarding                    http://77d87c5f.ngrok.com -> 127.0.0.1:3000
Forwarding                    https://77d87c5f.ngrok.com -> 127.0.0.1:3000
Web Interface                 127.0.0.1:4040
# Conn                        2
Avg Conn Time                 205.79ms
```

You can now try it out and type the URL that is given to you in your browser. You should see the same interface as before, meaning you can control your system from anywhere in the world!

One last word: be careful with this. Right now anyone can access this interface with the right URL, and now it is just connected to a relay that is actually not connected to anything, but don't do this if your whole alarm system is connected to the Raspberry Pi! In this case, you better put a solid login/password system on your server so that only you can access it.

How to Go Further

We built a Node.js-based application to automatically access the measurements coming from sensors. We also used a Node.js module to access the camera module of the Raspberry Pi. Finally, we displayed everything on a nice web interface. We also saw how to access our project from anywhere in the world using Ngrok.

You can go further with this project by including more sensors to the project. For example, you can add light level sensors and motion sensors. It is then quite easy to display the state of these sensors in the Node.js application.