Table des matières

[Thermostat documentation 1](#_Toc500863840)

[What is the purpose? 1](#_Toc500863841)

[Functions 2](#_Toc500863842)

[Basic services 2](#_Toc500863843)

[Extended services 2](#_Toc500863844)

[The thermostat can receive some external information to adjust temperature instruction: for instance, decrease temperature when you are out and have set your home is under alarm system. 2](#_Toc500863845)

[Architecture overview 3](#_Toc500863846)

[ 3](#_Toc500863847)

[Technical implementation 4](#_Toc500863848)

[Heating regulation 4](#_Toc500863849)

[Heating instruction remotely 4](#_Toc500863850)

[Hardware design 4](#_Toc500863851)

[Network connection 6](#_Toc500863852)

[Server components 8](#_Toc500863853)

[DIY instructions 9](#_Toc500863854)

[Build of material 9](#_Toc500863855)

[User guide 10](#_Toc500863856)

[Maintenance guide 10](#_Toc500863857)

[An open system 10](#_Toc500863858)

[Change than can be done easily 10](#_Toc500863859)

[Improvements that could be done 10](#_Toc500863860)

[Communication specifications 10](#_Toc500863861)

# Thermostat documentation

### What is the purpose?

* Increase your comfort by heating your house exactly as you want
* Make savings by heating your house only when you need
* On top of that be proud you did it by yourself

## Functions

### Basic services

Heating regulation algorithm adjusted to the environment provides a very stable temperature.

Heating instructions are defined on schedule based on weekly and daily calendars.

For instance: a weekly calendar to be used during the working periods and a daily calendar to be used when you spend holidays at home.

User expresses instruction as a couple (temperature,time) objective. For instance: instruction 20° at 8:30. The thermostat will start heating at the appropriate time depending on the inside and outside actual temperatures

User can at any time either with a remote controller

* switch between the calendars
* manually modify temperature instruction
* temporarily hold on the heating system

### Extended services

Even if the thermostat can work 100% autonomously its purpose is to be connected to provide additional services when connected to WIFI.

A Web application can be used for the user to monitor and control the heating. For instance, you will be able to increase the temperature remotely with your mobile on you way back from holiday.

The thermostat will provide a lot of information that could be used to better tune your installation and so increase your comfort and savings. Parameters can be modified remotely.

## The thermostat can receive some external information to adjust temperature instruction: for instance, decrease temperature when you are out and have set your home is under alarm system.

The clock of the thermostat is synchronized with the server and will automatically switch for the daylight saving time (summer time).

## Architecture overview



Internet WIFI

Router

Server

Tomcat Linux

WIFI Gateway

ESP8266

Micro Controller

Atmel ATmega

Boiler

## 

## Technical implementation

The core functions run on an Atmel ATmega micro-controller. It has been validated on the following Arduino platforms: Nano, Uno and Mega). Since code and parameters has been downloaded the Arduino and clock synchronized with the serer, it can run 100% autonomously. It communicates thru the serial link, eventually takes into account external information.

Parameters are initially written in the Eeprom and can be modified remotely and saved in the Eeprom.

### Heating regulation

Temperature is driven by a PID regulator. The PID parameters (KpPID,KiPID,KdPID, thresholdPID, PIDCyleDelay) are initially defined when downloading the code and can be adjusted either or remotely. When the thermostat is set in PID tuning mode it continuously sends information to the server that are useful to adjust the PID parameters to your equipment.

A timer parameter (hysteresisDelay) is used to avoid to switch the boilier on/off to fast.

Internet provides actual external temperature at your location.

### Heating instruction remotely

Even if temperature can be set manually, the purpose of the thermostat is to run in automatic mode.

The system will choose between choose between 4 different temperature instructions (for instance morning, day, evening, night) according to your need. Your need has to be defined in 1 weekly and 2 daily agendas. For each half an hour you have to select one of the 4 temperature instructions. The schedule has to be downloaded with the code but can be modified remotely and saved in the eeprom.

A 5th temperature instruction is defined and used when you are out of home to keep your house out of freeze.

Temperature instructions are expressed as targets (be at a temperature at a time). To decide when to start heating the thermostat will take into account the difference between the actual internal temperature and the target and the actual outside temperature. This regulation can be tuned to your need with 2 parameters (reactivity, sizeAnticipation) that you can modify remotely and save in the eeprom. Each hour instruction used one byte: the first half byte is used to select the instruction for the first half hour and the second half byte is used to select the instruction for the second half hour. For instance, 0x12 means select temperature 1 between 0 to 29mn ad temperature 2 between 30 to 59mn.

2 parameters (maximumTemperature, minimumTemperature) define the maximum and minimum temperature than can be set by the user. It can be modified remotely and save in the eeprom.

### Hardware design

#### Boiling connection

Boiling is connected to a NO relay. The heating is set on by closing the relay and of by opening it.

#### Power

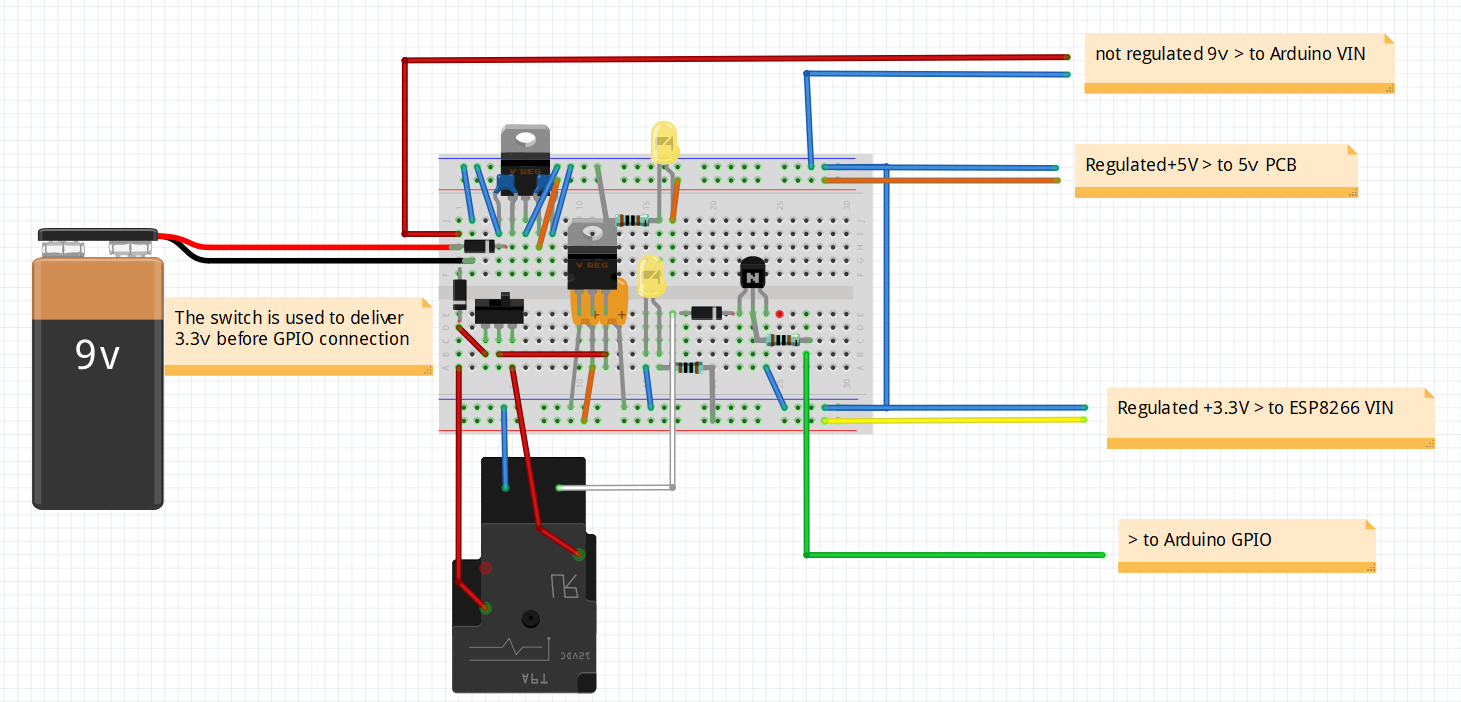
Power source can be delivered by any 9v DC 500mA not regulated. All components are soldered on a PCB. It delivers the original not regulated 9v power, a 5v and 3.3v regulated power. 3.3v is connected thru a NC relay to allow the Arduino to switch 3.3v On/Off. 2 diodes protect from wrong input connection.

9v powers the Arduino (VIN).

5v powers the 2 relays, the clock, the LCD, the LEDs and the infrared receiver.

3.3v power the ESP8266

The temperature sensor is powered by the Arduino 5v output.



#### Electronics

Arduino is protected from relays by NPN transistors and diodes.

ESP8266 RX (3.3v) is protected from Arduino TX(5v) by a voltage divider.

The design allows to replace easily all the active components.

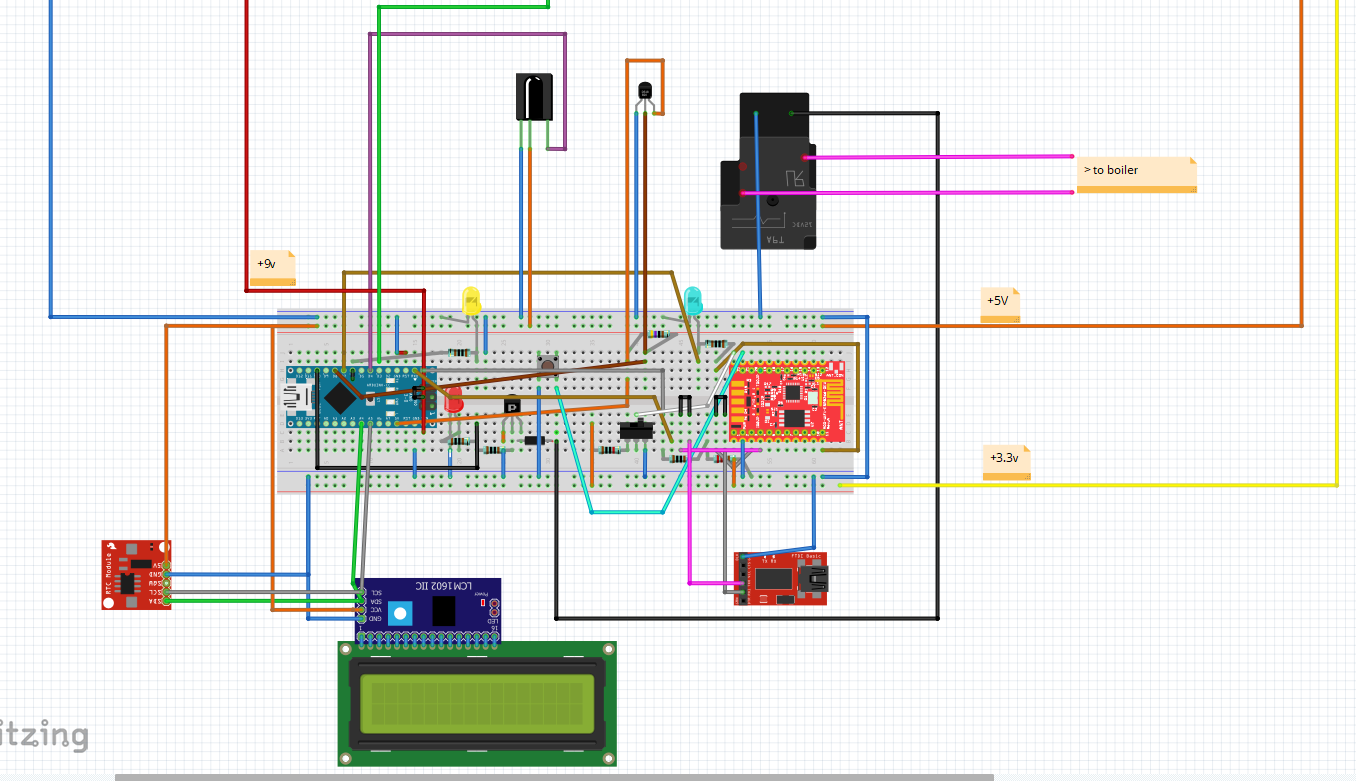
A push button allows to download ESP8266 program

2 connectors are used to switch the ESP8266 serial link from Arduino to the FTDI.

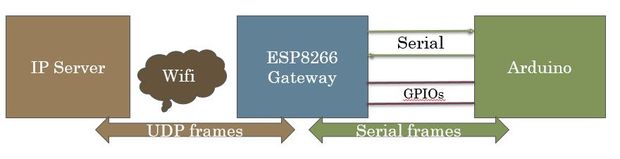
1 switch is used to set the ESP8266 in configuration mode

1 connector is used to set the Arduino in initialization mode

Arduino interacts with digital GPIO, serial link, One wire bus and I2C bus.



### Network connection



The network connection is made with a ESP8266 WIFI microcontroller. It is based on the [gateway description ‘instructables”](https://www.instructables.com/id/How-to-Make-Your-Own-WIFI-Gateway-to-Connect-Your-/) . The following changes have been made from this description: some useless GPIOs for this project are not used and the Arduino and ESP8266 are soldered on the same PCB.

This module is connected from one side with the serial link from the other side to IP network with the Wifi. It acts as a black box. Data packets coming from the serial link are sent to an IP/Udp port and vis et versa. It can transfer either ASCII and binary data.

You just have to set your own configuration (IP, WIFI ...) once the first time you will power on the Gateway. This gateway configuration can be changed locally with USB TTY interface (see maintenance guide). Configuration is saved in eeprom.

A simple protocol is used between Server and Arduino as follow

* end to end 8bit CRC to check frame integrity
* send and receive frame numbers to check for missing frame
* frame sent can request or not request an acknowledgement from receiver

Both Server and Arduino initiate a dialog

* send information without being requested for
* send a request
* send a response to a request

## Server components

Server runs Linux (developed with Ubuntu 14.04), mySQL server (developed with 5.5) and Tomcat7.

Batchs are in charge of communication with the Thermostat.

It reads “meteo” table to look for the actual outside temperature and prepare to send to the thermostat

It reads “ind\_desc” table to find parameter’s values that eventually need to be send to the thermostet

It writes in “ind\_value” table parameter’s values and mesurments sent by the thermostat

J2EE application is in charge of human interface. It reads and updates database. It also send command to the thermostat going thru batchs.

Thermostat

Read

Write

Request

Response/Information

Web application

J2EE

meteo

Ind\_value

Ind\_desc

Batchs

Java

2EE application are in charge of exchanges with the user. It reads and updates database. It also send command to the thermostat going thru batchs.

## DIY instructions

The first step is to gather all the parts you will need.

### Build of material

**You will need these main components**

*2 x micro-controllers*

* + 1 x Arduino - I chose a Nano 3.0 - you can find some at around 2.5$ (Aliexpress)
  + 1 x ESP8266 - I [chose -ESP8266-DEV Olimex - at 5.5€](https://www.olimex.com/Products/IoT/MOD-WIFI-ESP8266-DEV/open-source-hardware)

*1 x temperature sensor DS1820*

* I chose a waterproof one - you can get 5 for 9€ (Amazon)

*1 x double relay module (0 command)*

* I chose SONGLE SRD-05VDC - you can find some at 1.5€ (Amazon)

*1 x I2C LCD 2x16 characters*

I already had one - you can find some for less than 4$ (Aliexpress)

*1 x I2C*DS1307 *Real Time Module with CR2032 battery*

* I already had one - you can find some for less than 4$ (Aliexpress)

*you can find for a few euros*

*1 x Infrared receiver*

* I chose AX-1838HS you can find 5 for 4€

1 x FTDI

**And some few stuff**

5 x LED

9 x 1K resistors

1 x 2.2K resistor

1 x 4.7K resistor

2 x NPN transistors

4 x Diodes

2 PCB breadboard

2 x 3 pins switches

Some connectors and wires

Of course, soldering iron and tin to do it.

### Build the power sources

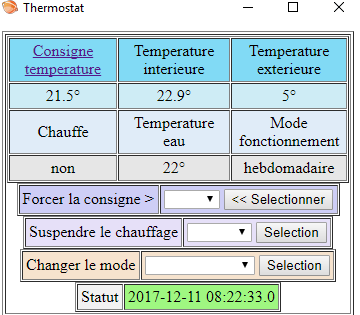
This fritzing file describes what to do.

As usual it is better to build the power sources with a breadboard.

Check it delivers a constant 5v and 3.3v even with a load (100 ohms resistors for instance).

You can now solder all the components on a breadboard PCB as below.

## User guide



## Maintenance guide

## An open system

### Change than can be done easily

Replace internet weather forecast by a DIY external temperature sensor.

### Improvements that could be done

### Communication specifications