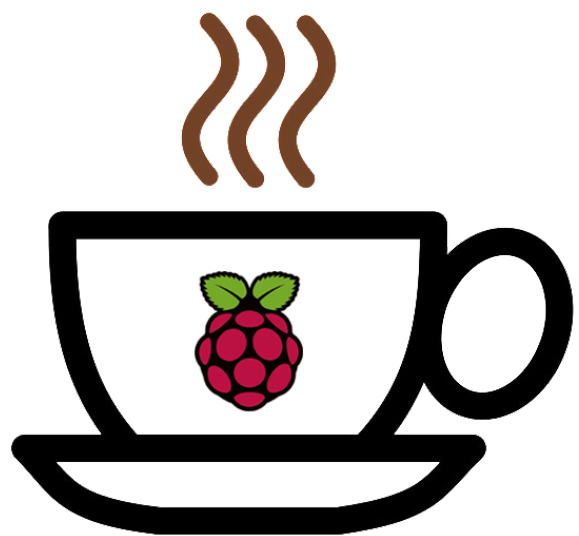


piCoffee



Description

The aim of this project is to recycle an old coffee machine to control it through the smartphone. This is possible by changing the wiring diagram inside the device, connecting it to a Raspberry that will act as a web server and will wait for our indications.

The advantages that this type of modification brings are:

1. An energy saving: the machine in question is quite dated and, leaving it switched on, it would consume a lot of power. With the change made, the possibility is given to make coffee with maximum efficiency and in a short time, as the sequence of operations to be performed is automated.
2. The planning of the preparation of a coffee: through **crontab** and the interface provided, the user can program the delivery of the mixture.

Tools and development environments used

- Relay relay 5V - 230V
- Monostable relay
- Raspberry Pi 3 Model B
- Saeco Nina cappuccino Type Sin026X
- Cable kit Elgoo
- Atom
 - IDE for HTML and PHP
- Putty
- FileZilla
- Bash Debian

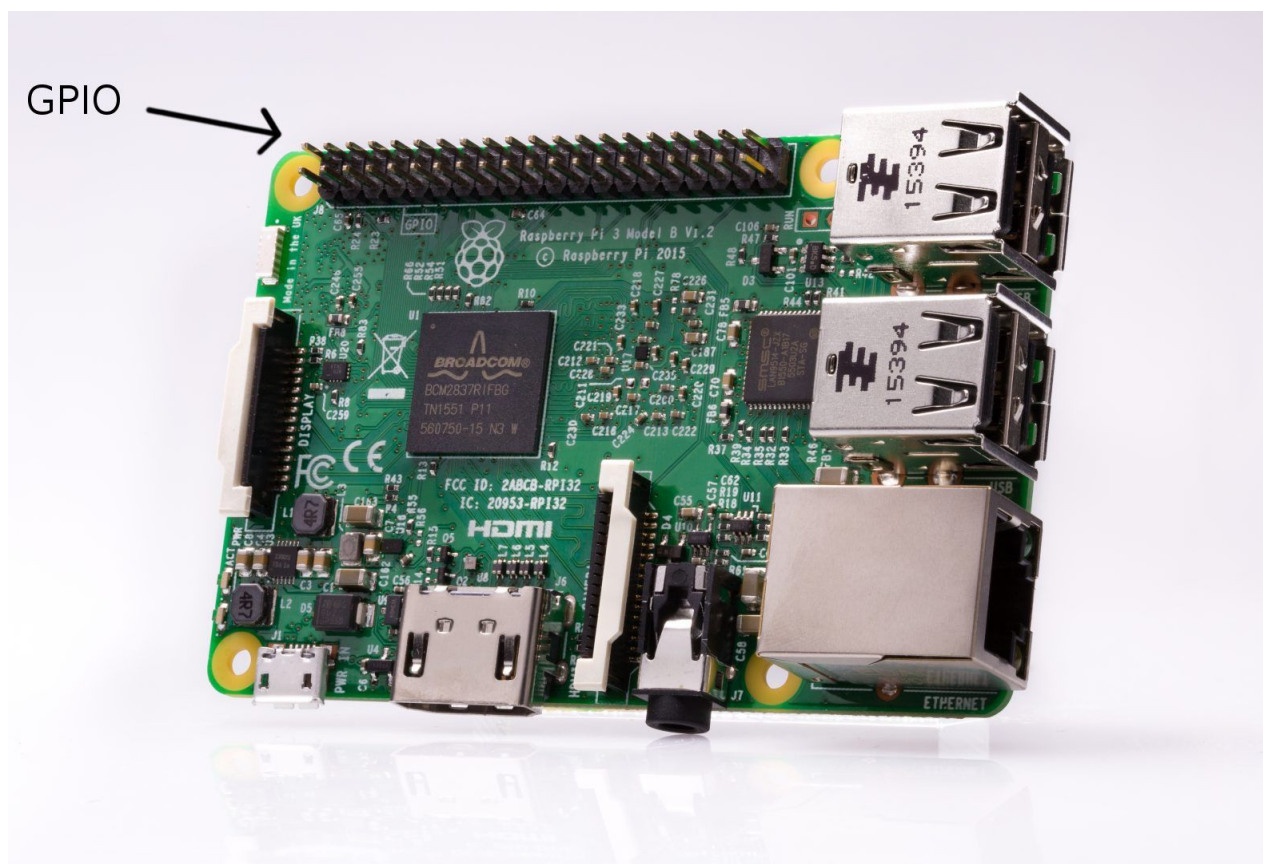
What is the Raspberry Pi

The Raspberry Pi is a single- computer board developed in the United Kingdom by the Raspberry Pi Foundation. Several OSs can be installed in it, the most common being Raspbian, a Debian version modified specifically for the device.

The latest version released in March 2018 has a cost of only € 37.

Its uses can be the most diverse: as economic computers in developing countries, as a media center, or in general, for do-it-yourself projects. In my case it will serve as a command center: it will receive requests from the connected device and transmit power to the coffee machine if necessary.

The specific hardware that connects electronic and IT is the line of PIN GPIO (General Purpose Input / Output). In fact, these PIN types can be set to send or receive current. The sending will be used to turn the machine on, while receiving it to understand when the boiler is sufficiently hot.



Electrical Design

Normal operation

The machine starts making coffee when the central knob is turned to the right. Behind it, in fact, there is a switch that activates the circuit and starts the water pump, which will pass through the boiler and finally into the coffee dust, where the infusion will take place. An LED above the knob shows when the boiler is hot enough to make coffee. The indicator lights up when the thermostat reaches the useful temperature.

Modified operation

Power supply

Power is supplied via two electrical plugs. One will supply power to the Raspberry and one to the coffee machine. The machine is not always on, but will only be powered when the Raspberry decides it. The electrical phase is normally interrupted by the *computer*, but when you decide to turn it on, the PIN will supply voltage and close the circuit, starting the device. It was not possible to feed the whole thing from a single plug because the mini-computer would not be able to supply the necessary current to the coffee machine, even with the relay support.

Water pump

The water pump must only work when the boiler is hot. PIN 2 is connected to the relay, which in turn is connected to the pump, replacing the knob. The latter is activated with the command `gpio write 2 0` and deactivated by `gpio write 2 1`. Between one command and the other the function `sleep(duration_in_seconds)` is used to stop the execution of the PHP code and consequently to supply the right amount of coffee, since the switch-off command is not performed immediately, but only after the time defined by *duration*, which was previously chosen by the interface.

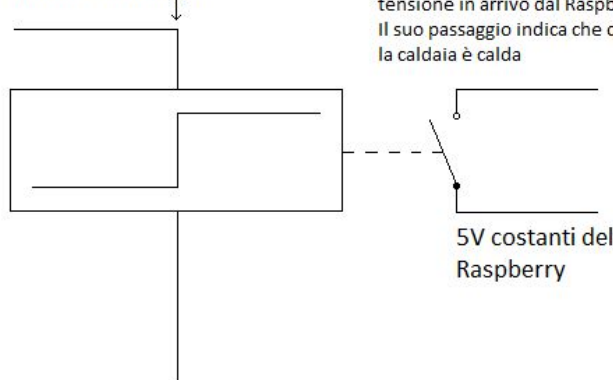
Boiler / Thermostat

The boiler is controlled by a thermostat and when the latter detects the appropriate temperature, it turns on a LED next to the knob. Getting the Raspberry to receive the signal was not easy, since it can not receive 230V, but only at 5V.

To remedy the problem, I connected a second relay to the Raspberry. On one side I connected the cable that made the light illuminate. When voltage goes from here, the relay will energize and close the circuit, turning on the led. At the same time, by closing, it will let pass the current coming from PIN 0, always active and used only for this task; in this way the electric energy will go directly to PIN 1 that is listening.

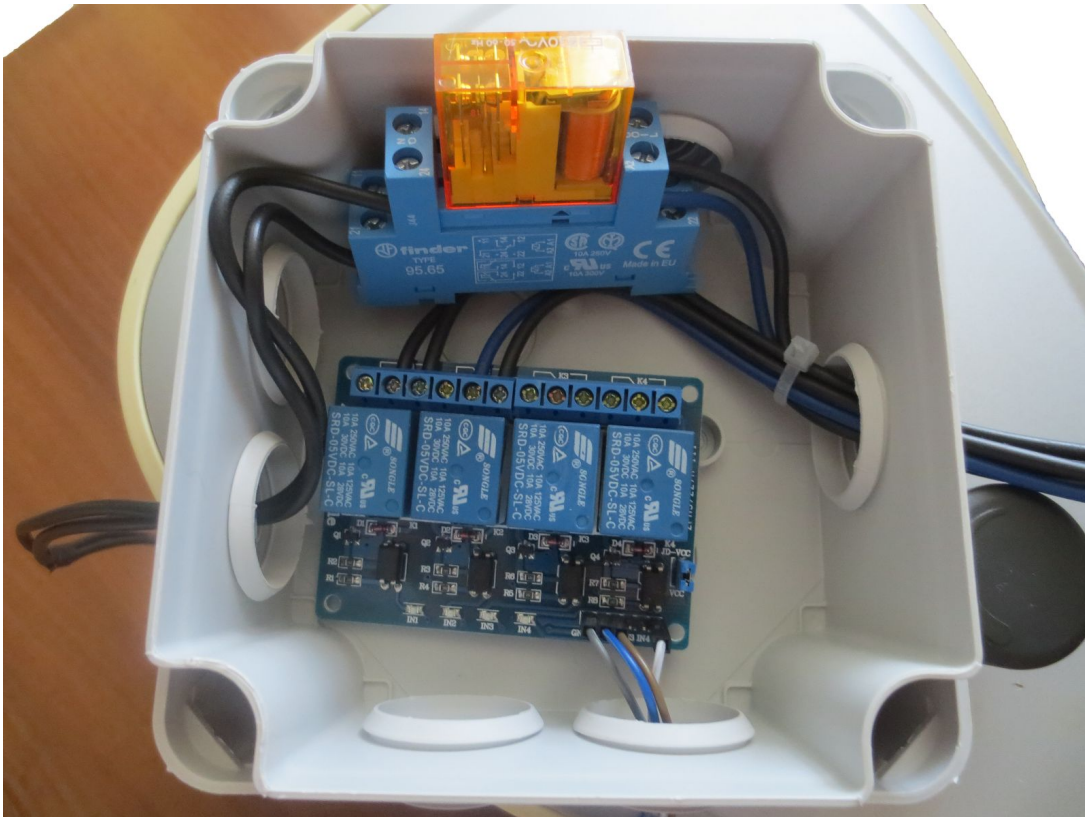
When PIN 1 receives 1, then the boiler is ready.

la spia luminosa è collegata in parallelo al relè. Quando si accende passa corrente anche qui



Knob

The new knob function is about safety. Turning it to the right, the coffee machine will be available to all our requests. If for some reason we should run into some malfunction (Eg the device does not stop making coffee, the boiler heats up indefinitely, etc.), we can place the knob in the middle. In this status, any action in progress is ignored, such as coffee delivery. In the meantime, the code will continue its execution, then either wait for its end and reposition the knob, or, in the worst case, the plugs will come loose. Physically, the coffee machine is as if it is switched off, in the same way as when it was turned off by the button on the right side of the casing.



Relay in position

Software Design

WiringPi

WiringPi is a library of PIN GPIO access written specifically for the various Raspberry models. The software includes a series of shell-executable commands (**gpio**) that allow you to check GPIO PINs.

Eg.

```
gpio mode 1 out # Pin number 1 is set to output mode.
```

```
gpio write 1 1 #Pin number 1 is live.
```

At the boot of the Raspberry are executed some commands that are used to set the PINs correctly and make them ready for use by the PHP pages. The file that allows this is `rc.local` which is in the `/ etc / directory`.

Command sequence:

```
gpio mode 0 out # Set PIN 0 in output mode
gpio write 0 1 # Set PIN 0 in voltage
gpio mode 3 in # Set PIN 3 in input
mode gpio mode 2 out # Set PIN 2 in mode output
gpio write 2 1 # Put the PIN 2 in voltage
gpio mode 1 out # Set the PIN 1 in output mode
gpio write 1 1 # Put the PIN 1 in tension
```

Materialize

Materialize is a responsive front-end framework based on the Material Design, the style designed by Google. It can be considered a valid alternative to Bootstrap, another responsive framework. Both, in fact, are based on the use of the 12-column grid system.

The framework provides a CSS file and a JavaScript file with ready-made styles and animations for use on your pages.

Crontab

Crontab is a command that can be installed on Linux operating systems that allows the planning of commands, which can be sent running periodically. The single event in the crontab is called cronjob. My intent is to give the user the possibility to program the coffee output. For this to happen, it was essential to modify the privileges of the user *www-data* to allow him to launch the command *php-cgi*, which is necessary to run a PHP page with GET parameters from the command line. In particular, I had to add *www-data* to the list of *sudoers*, ie users who can use the command *sudo* without needing to enter the password. Safety in this case is not a priority as the coffee machine can only be used in a local network.

```
* * * * * sudo php-cgi /var/www/html/comandi/on.php duration = 22
```

Example of cronjob

```
.----- [m]inute: minuto (0 - 59)
| .----- [h]our: ora (0 - 23)
| | .----- [d]ay of month: giorno del mese (1 - 31)
| | | .----- [mon]th: mese (1 - 12) OPPURE jan,feb,mar,apr...
| | | | .---- [w]eek day: giorno della settimana (0 - 6) (domenica=0 o 7) OPPURE sun,mon,tue,wed,thu,fri,sat
| | | | |
* * * * * comando da eseguire
```

How the timeline is structured in a cronjob command


To make the timeline understandable to everyone, I added a library called **cRonstrue**, which provides JavaScript functions that transform the numeric string into a readable string.

Ex.

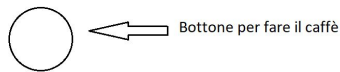
```
Cronstrue.toString ("0 23 * * 1,5", { locale: "it" })
// The string value is: "At 11pm, Monday and Friday"
```


How all this work together

User will have 3 available interfaces: fast coffee, programming and options

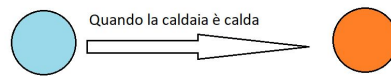
 piCoffee
Caffè veloce | Programma il caffè | Opzioni

Spiegazione del pulsante



Scegli il caffè che più ti piace

corto  lungo  Switch = 2 radio button




Spiegazione del pulsante

Fast coffee:

- The user can press the button to deliver the coffee efficiently, as all the operations are carried out in the quickest sequence
- Through a switch he can choose whether to make an espresso or a long coffee a small icon provides the status of the boiler
 - Light-blue and firm: the boiler is cold
 - Red and “flashing”: the boiler is hot

Planned Coffee:

 piCoffee
Caffè veloce | Programma il caffè | Opzioni

Scegli il caffè che più ti piace

corto ☐ lungo ☐

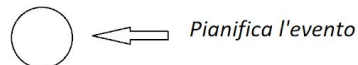
Programma il caffè in un singolo giorno ☐ Programma il caffè ogni settimana nello stesso giorno ☐

↓

☐ Lunedì
☐ Martedì
☐ Mercoledì
☐ Giovedì
☐ Venerdì
☐ Sabato
☐ Domenica

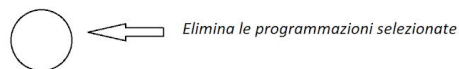
Form che apre un calendario e fa scegliere la data

Form che apre un orologio e fa scegliere l'ora



Pianificazioni in corso

☐ Elenco delle pianificazioni in corso



- Ability to choose the type of coffee
- The schedules are different, the first program coffee a day (with renewal) annual , while the other is repeated every week on selected days / o
- The time and day are chosen through a special form of Materialize, which returns a string.

Options

- Allows only switching on (and then switching off). It may be necessary if you have to make a lot of coffee, so the boiler will not have to be switched on every time, thus avoiding cooling.
- A message reminds you to turn off the machine. Leave it on could consume a lot of power

Accensione e spegnimento

Accendi la caffettiera

Spegni la caffettiera

Messaggio che ricorda di spegnere la macchina del caffè

NB

- All requests to the server are made via AJAX because it gives the possibility to obtain information in real time without reloading the page (Eg if the boiler is in temperature)
- The sections "*Fast coffee*", "*Program the coffee*" And "*Options*" are part of the page **index.php**

tree

```
/var/www/html
├── commands
│   ├── accendi_spegni.php
│   ├── controlla_acc.php
│   ├── eliminazione.php
│   ├── leggi_caldaia.php
│   ├── on.php
│   ├── program_elenco.php
│   └── programmazione.php
├── cronstrue
│   ├── dist
│   │   ├── cronstrue-il8n.js
│   │   ├── cronstrue-il8n.min.js
│   │   ├── cronstrue.js
│   │   └── cronstrue.min.js
│   ├── il8n
│   │   ├── locales
│   │   │   ├── en.d.ts
│   │   │   └── it.d.ts
│   │   └── locales
│   └── locales
│       ├── en.d.ts
│       └── it.d.ts
├── img
│   ├── favicon.png
│   └── logo1.png
├── index.php
├── materialize
│   ├── css
│   │   ├── materialize.css
│   │   └── materialize.min.css
│   ├── fonts
│   │   └── roboto
│   └── js
│       ├── materialize.js
│       └── materialize.min.js
├── LICENSE
└── README.md
```

The pages selected in yellow are those written or edited by me, the others are the two libraries I have used. I have omitted the files that I did not use directly.

The pages contained in "commands" launch a command in the Raspberry Pi shell through the PHP system (command, output) function. In the first parameter I have always written a read and write operation of the GPIO.

Eg.

```
Gpio read 3
```

```
gpio write 0 1
```

The name of the files is quite explanatory:

- **accendi_spegni.php**, turn on the coffee machine if it is off or off if it is turned on
- **controlla_acc.php**, check if the device is on
- **eliminazione.php**, eliminates events scheduled
- **leggi_caldaia.php**, returns 0 if the boiler is cold, 1 if hot
- **on.php**, prepare the coffee and turn on the machine if off
- **program_elenco.php**, return the list of scheduled events
- **programming.php**, program an event



Coffee machine completed

Example of operation

After being connected to the Raspberry website, through its IP address, you choose an espresso coffee from the *switch* and touch the button with the symbol of the cup.

At that moment an AJAX request starts at the on.php page, with a GET parameter, called *duration*. If you chose espresso coffee, the variable will be 22 seconds, while the long one is 35 seconds.

```
url ="commands / on.php? duration =" + encodeURIComponent(dura_coffee);  
richiesta.open ("GET",url,false) // INDUCTION
```

Url and function used in the request. *duration_coffee* is worth 22 in this example

At this point the on.php page turns on the coffee machine and therefore starts to heat up. After this command starts a while loop, which repeats every 3 seconds and which has as a condition `gpio read 3`, that is until the thermostat does not signal that the boiler is in temperature.

```
while (system ("gpio read 3") ==0) { // As long as the server response is negative (ie  
0), the  
                                // loop repeats itself. In this way it prevents the  
pump  
                                // from being activated immediately  
    sleep (3);  
}
```

When the program leaves the cycle, start the water pump. The switch-off command, however, is preceded by the function `sleep(time in s)`. The parameter is precisely the duration, decided beforehand with the *switch*.

After the *sleep* the pump is switched off. If the machine was already on before touching the symbol of the coffee cup, leave it in the same state, otherwise it will switch it off.

Meanwhile, the interface communicates that coffee is being prepared and an upload animation is added. Furthermore, a symbol shows when the boiler is in temperature.

Once the coffee has been dispensed, a message of successful delivery appears.

Sources

- <https://crontab.guru/>
- <http://bradymholt.github.io/cRonstrue/>
- https://en.wikipedia.org/wiki/General-purpose_input/output
- <https://www.raspberrypi.org/documentation/usage/gpio/>
- <https://it.wikipedia.org/wiki/Crontab>
- <https://materializecss.com/>
- <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>
- <https://www.raspberrypi.org/documentation/linux/usage/cron.md>
- <https://stackoverflow.com>
- <http://php.net/>