

IRRIGADOR INTELIGENTE

Alice Fazzolino 12/0108747

Programa de Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
email: afazzolino@gmail.com

Renato Cesar da Silva Agnello 12/0053896

Programa de Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
email: ragnello19@gmail.com

1. RESUMO

Sistema de irrigação inteligente capaz de monitorar o clima de uma determinada região via internet e via sensor por meio da Raspberry pi e através dessa informação irrigar ou não um jardim e/ou plantação.

2. INTRODUÇÃO

De acordo com o Ministério do Meio Ambiente [1], a agricultura consome mais de dois terços da água doce utilizada no planeta. Isso porque na maioria das vezes há um mau aproveitamento da água, provocando um alto desperdício da mesma. As previsões futuras da Organização das Nações Unidas [2] mostram que, se não houver uma economia significativa, em 2050 mais de 45% da população mundial estará vivendo em países que não garantirão a cota mínima diária de 50 litros de água por pessoa. Com isso, é possível compreender a importância de desenvolver sistemas capazes de oferecer um melhor aproveitamento da água, sem que haja desperdícios desnecessários.

Então, com esse objetivo, a essência deste projeto é colaborar com usuário no processo de irrigação do solo quando o mesmo é necessário, realizando isso de modo automatizado e com exatidão, não permitindo que o solo seja irrigado desnecessariamente e assim evitando o desperdício de água.

É de suma importância salientar que cada tipo de plantação necessita de uma pesquisa detalhada, uma vez que o grau de umidade pode ser distinto para cada uma, como também pode haver uma variação conforme o tipo de solo e outros. O mesmo acontece com o clima da região, sendo necessário um estudo prévio antes da implementação do sistema embarcado. Contudo, o projeto não entra nesses assuntos, visto que este é de caráter universitário de um curso de Engenharia Eletrônica e se enfatiza no desempenho entre o computador (Raspberry), o sensor de temperatura/umidade e o site <https://openweathermap.org/>.

No presente projeto foi construído com poucos componentes um sistema capaz de irrigar jardins e/ou plantações de diferentes portes. Utilizando-se de uma Raspberry pi 2 foi possível acessar um site

(<https://openweathermap.org/>) que dá informações sobre o clima de diversas cidades/países, tanto do clima atual como previsões futuras. E com isso, juntamente com o sensor de temperatura/umidade DHT11, as informações são analisadas e de acordo com as mesmas o circuito permite a irrigação ou não.

Também é importante ressaltar que especificações como tempo de irrigação, quantidade de irrigação, tempo de intervalo entre irrigações e outros, basta realizar a edição do código do sistema.

3. DESENVOLVIMENTO

3.1 DESCRIÇÃO DO HARDWARE

O projeto geral de Hardware está descrito na Figura 1. Os principais componentes do diagrama de blocos abaixo serão discutidos a seguir.

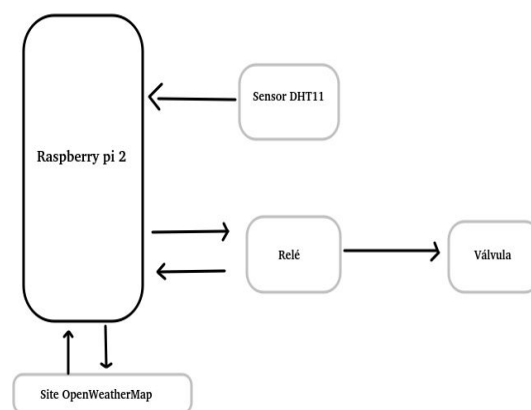


Figura 1: Diagrama de Blocos

A Raspberry Pi 2 utilizada possui 26 GPIOs (General Purpose Input/Output) e as mesmas operam numa tensão de 3.3 V para o nível alto e 0V para o nível baixo. Isso pode acabar limitando as aplicações, pois a maioria dos componentes e circuitos operam com o nível de tensão de 5V, como por exemplo o relé. Uma forma de solucionar essa limitação é utilizar os pinos da Raspberry apenas para

controlar os circuitos que acionarão dispositivos externos em vez de acioná-los com a própria Raspberry.

Na Figura 2, pode ser visto que como a bobina do relé necessita de uma tensão de 5V para funcionar, o GPIO aciona o MOSFET IRF540 canal N, que funcionando como uma chave, liga e desliga o relé, e o relé realiza o controle da válvula. Quando o pino GPIO estiver em nível baixo (nível 0), o relé estará aberto (não deixando a válvula ser energizada) e a válvula fechada, ou seja, a água não fluirá. E quando o pino GPIO estiver em nível alto (nível 1), o relé fechará (fará com que a válvula seja energizada) e a válvula abrirá, ocorrendo a irrigação.

O diodo serve apenas para garantir que nenhuma corrente flua em sentido oposto ao correto.

Já o relé controla a alimentação de 24V para a válvula solenóide. E os leds são utilizados apenas como indicadores, para saber se o circuito está recebendo energia ou para saber se válvula está aberta ou fechada e etc.

O sensor DHT11 e o site <https://openweathermap.org/> foram usados simultaneamente para obter informações de temperatura, umidade e nebulosidade. O sensor utilizado foi o DHT11, mas o ideal seria utilizar o sensor de umidade de solo Higrômetro para fazer a leitura da umidade do solo e assim saber se é necessário efetuar a irrigação ou não. Já o site OpenWeatherMap.org foi utilizado para ter a informação do clima na região em tempo real e previsões futuras. A pergunta que pode surgir é porque utilizar-se de dois meios diferentes para obter a mesma informação. E a resposta é: Economia de água. Pois pode acontecer do sensor de umidade do solo apontar que o solo está seco, ou seja, efetuar a irrigação, e dentro de poucas horas chover. Ou seja, estaria molhando a plantação repetidamente sem necessidade. Então, o site é para garantir que isso não aconteça, se na previsão apontar que choverá dentro de poucas horas, a irrigação não será realizada.

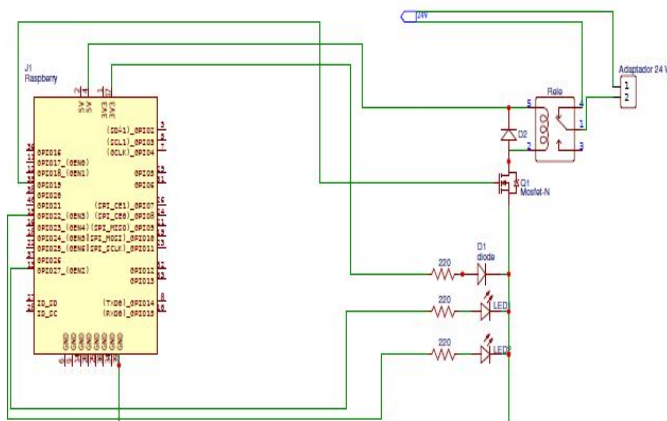


Figura 2- Projeto geral de Hardware

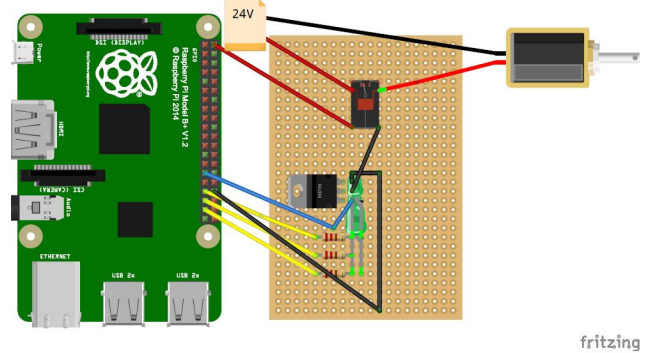


Figura 3- Esquema do circuito

3.1.1 MATERIAIS

Materiais	Quantidades
Raspberry pi	1
Protoboard	1
MOSFET IRF540N	1
Sensor DHT11	1
Relé 5V	1
Válvula de vazão de água (solenóide) 12VDC	1
Fonte de notebook 20V	1
Diodo 1N4001	1
Resistor de 4.7K	1
Jumpers	--

3.2. DESCRIÇÃO DE SOFTWARE

3.2.1 FLUXOGRAMA

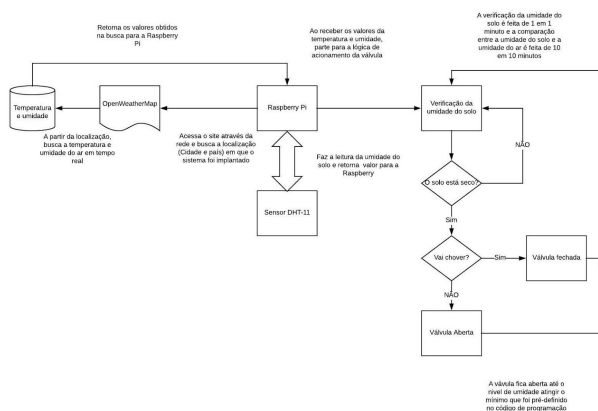


Figura 10 - Fluxograma do código

Pode-se observar no fluxograma que o centro de tudo é a Raspberry, pois ela faz o controle do sensor, a conexão com a rede para buscar os dados requisitados e libera a energia nos pinos GPIO necessária para o funcionamento do relé e, por consequência, o acionamento da válvula.

A Raspberry se conecta com o site da *Open Weather Map*, busca os dados desejados e por fim armazena esses dados para fazer a comparação mais a frente.

Após a obtenção dos dados, faz a leitura da umidade do local através do sensor que retorna a temperatura e a umidade como parâmetros, armazenando também essas informações.

Por fim a Raspberry usa os dados armazenados para fazer a lógica de acionamento do relé e por conseguinte da válvula. Ela verifica a umidade do solo e faz a comparação com a previsão do tempo.

3.2.2 IMPLEMENTAÇÃO

O código foi desenvolvido para fazer o controle de abertura e fechamento da válvula automaticamente. Ele busca os dados em tempo real da temperatura e umidade do local no qual o sistema foi implementado, fazendo a busca dessas informações através da rede (internet).

Após o recebimento desses valores, a Raspberry faz a leitura da umidade do solo através do sensor DHT-11 (sensor de temperatura e umidade). Com ambos os valores armazenados, é feita uma comparação entre eles e, através da lógica implementada no código, o sistema decide se a válvula deve ser acionada ou não.

Com isso, seguem as descrições e explicações do código:

```

#include "gpio_sysfs.c"
#include "gpio_sysfs.h"
#include <stdio.h>
#include <stdlib.h>
#include <threads.h>
#include <unistd.h>
#include <signal.h>
#include <fcntl.h>
#include <time.h>
#include <wiringPi.h>

// Arquivo de acesso a porta serial para Raspberry Pi 3
#define TTY "/dev/ttyS0"

// Definição de variáveis globais
#define HIGH 1
#define LOW 0

#define pino_rele 17
#define pino_sensor 26

int leitura = 0;
float temp_sensor = 0;
float umid_sensor = 0;
char api_key_openweathermap[35];
char cidade[15];
char pais[15];

```

Figura 11 - Inicialização das bibliotecas e variáveis globais

```

void le_info_clima(char *p_api_key_openweathermap, char *p_cidade, char *p_pais){
void main()
{
    wiringPiSetup();
    pinMode(pino_rele, OUTPUT);
    pinMode(pino_sensor, INPUT);

    pthread_t a_thread;

    #define variáveis globais
    api_key_openweathermap = "be6d6c3759bcb236fa4e9a312c10b45" #coloque aqui sua api-key da OpenWeather
    cidade = "London" #coloque aqui o nome da cidade desejada, sem acentos para nomes com espaço, substitua o espaço por
    pais = "uk" #coloque aqui a sigla do país em que a cidade esta (para o Brasil, a sigla é br)

    def le_info_clima():
        global api_key_openweathermap
        global cidade
        global pais

        url_http_req = "http://api.openweathermap.org/data/2.5/weather?q="+cidade+"&units="+pais+"&appid="+api_key_openweathermap
        dados = requests.get(url_http_req).json()

        #Informações desejadas do JSON
        temp_atual_kelvin = dados["main"]["temp"]
        umidade = dados["main"]["humidity"]

        temp_atual_celsius = int(float(temp_atual_kelvin) - 273.15)
        umidade_atual = float(umidade)

        return temp_atual_celsius, umidade
    
```

Figura 12 - Escopo da função na qual se obtém as informações do clima através da rede e a função “main”, contendo algumas variáveis utilizadas

```

while(True):
    try:
        #Leitura do sensor
        (temp_sensor, umid_sensor) = Adafruit_DHT.read_retry(sensor, pino_sensor)

        #Requisita informações da API e faz parser do json, filtrando as informações desejadas (temperatura, umidade e pressão)
        (temp, umidade) = le_info_clima()
        leitura = int(leitura) + 1

        temperatura_str = str(temp)+" C"
        umidade_str = str(umidade)+"%"

        temp_sensor_str = str(temp_sensor)+" C"
        umid_sensor_str = str(umid_sensor)+"%"

        #Escreve as informações lidas e a cidade referida
        print "Leitura: "+str(leitura)
        print "Cidade: "+cidade+", "+pais
        print "Temperatura atual: "+temperatura_str
        print "Umidade relativa do ar: "+umidade_str
        print ""

        print "Temperatura Sensor: "+temp_sensor_str
        print "Umidade Sensor: "+umid_sensor_str

        if umidade < 90:
            GPIO.output(pino_rele, GPIO.HIGH)
            time.sleep(5)
            GPIO.output(pino_rele, GPIO.LOW)
        else:
            GPIO.output(pino_rele, GPIO.LOW)
    
```

Figura 13 - Loop infinito contendo que faz a varredura do clima pelo sensor e compara com as informações obtidas na rede para decidir se faz o uso da válvula ou não para a irrigação do plantio

3.2.3 REGISTRO E OBTENÇÃO DA API

Para obter a API e utilizar os dados climáticos de diversas localizações, primeiramente deve-se fazer o cadastro no site da *Open Weather Map* (*openweathermap.org*).

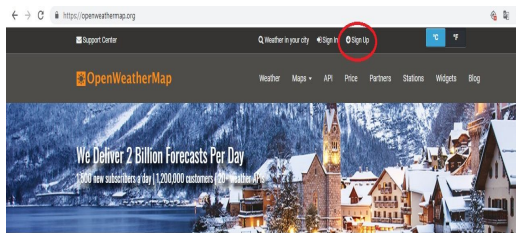


Figura 7 - Página inicial do site da *Open Weather Map*. Para fazer o cadastro basta clicar em *sign up*, como indicado em vermelho, e seguir os passos

Após a conclusão do cadastro, será direcionado diretamente para a página pessoal com diversas opções. Para obter sua chave API, basta clicar em “API Keys” e será direcionado diretamente para a página de suas chaves API.

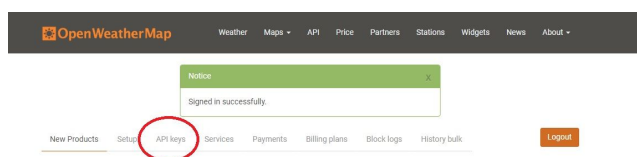


Figura 8 - Página principal, na qual o site te direciona ao término do cadastro.

Ao clicar em “API Keys”, você verá que por padrão o site já gera uma chave assim que terminar o cadastro e ela está com o nome de “API_Key”. Se quiser, poderá gerar quantas chaves quiser, dependendo do número de aplicações de seu sistema.

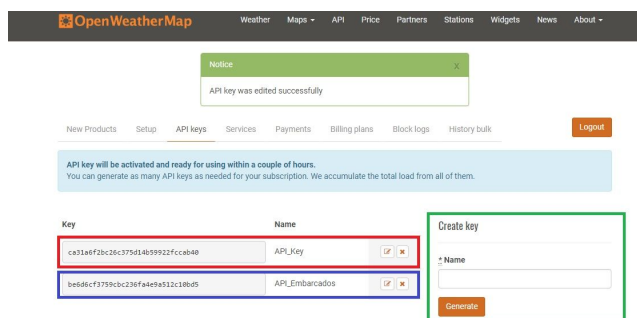


Figura 9 - Página das chaves API. Ela já vem com uma chave padrão (em vermelho) que pode ser utilizada ao terminar seu cadastro. Podem ser geradas outras chaves no campo *Create Key* (em verde). Só é necessário escolher o nome da chave e clicar em *Generate*. Após isso ela será adicionada na lista de suas chaves API. Como exemplo, podemos ver a nova chave gerada (em azul), nomeada de “API_Embarcados”.

Após a realização dos passos anteriores e obtendo a chave API é possível a utilização dela no código.

4. RESULTADOS

Os resultados obtidos foram os mesmos esperados e propostos no ponto de controle 1, quando se fala dos resultados na parte de obtenção de dados e controle do sistema. Podemos observar na Figura 11 que ao

pré-definirmos a cidade, conseguimos buscar e obter a temperatura e a umidade da cidade/país que desejamos.

Também podemos observar que a leitura do sensor funcionou perfeitamente como esperado, pois podemos ver a comparação feita entre os dados obtidos na rede (cidade de Londres, UK) e os dados capturados pelo sensor (Brasília, Brasil).

Houveram algumas mudanças com relação ao código e com o protótipo, por conta das dificuldades encontradas no decorrer do processo de execução.

Primeiro com relação aos componentes, queimou-se uma válvula, fazendo com que tivéssemos que comprar uma nova e aguardar a entrega. Com esse problema, ocorreram atrasos na montagem do protótipo, fazendo com que ele não atingisse o nível de montagem que era capaz de atingir.

Por fim, o código ainda será aperfeiçoado para a entrega do projeto final.

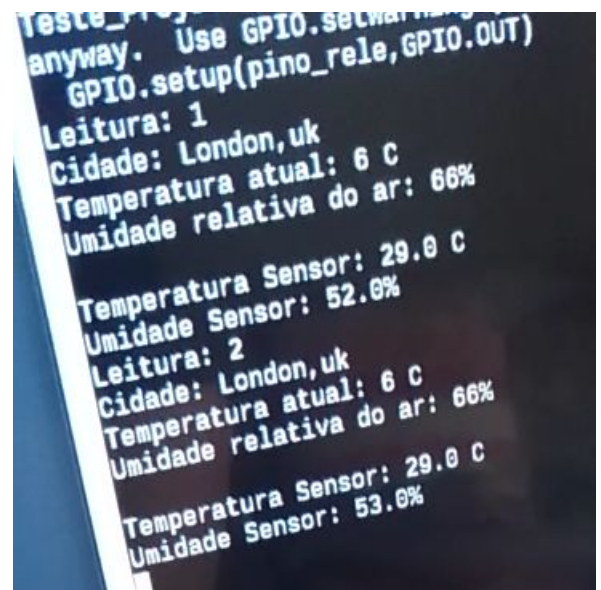


Figura 11 - Imagem do teste de obtenção dos dados do site através da rede

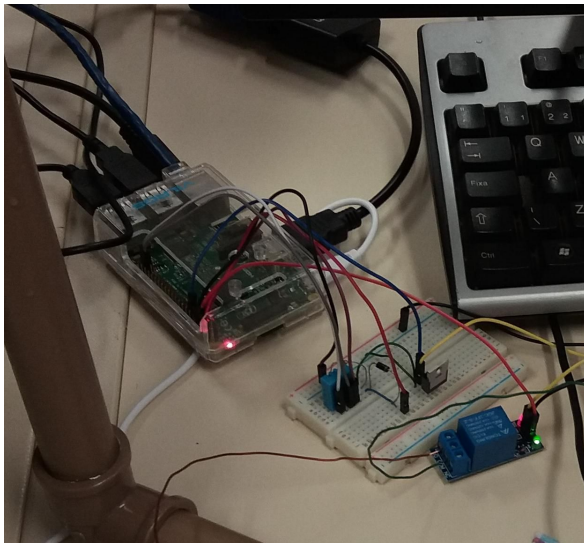


Figura 12: Montagem na protoboard

5. CONCLUSÃO

Com a utilização da Raspberry pi 2 para realizar o constante monitoramento do clima da região através do site <https://openweathermap.org/> e com os dados obtidos a partir das medições do sensor de temperatura/umidade DHT11 foi possível criar um sistema embarcado de irrigação capaz de diminuir o desperdício de água em jardins ou plantações de pequeno ou grande porte.

Como o projeto é desenvolvido na matéria Sistemas Embarcados durante o período letivo, o tempo é relativamente curto para se aperfeiçoar o mesmo tanto esteticamente como na questão da lógica de programação. No entanto, o projeto foi bastante proveitoso para a dupla, pois foi possível aprender um pouco sobre as infinitas aplicações deste fascinante computador que é a Raspberry Pi.

6. REVISÃO BIBLIOGRÁFICA

[1] Ministério do Meio Ambiente - Água. Disponível em: <<http://www.mma.gov.br>>. Acesso em: Dezembro de 2018.

[2] Organização das Nações Unidas no Brasil - A ONU e a Água - Disponível em: <<https://nacoesunidas.org/acao/agua/>> Acesso em: Dezembro de 2018.

[3] FINIO, Ben. Raspberry Pi Controlled Irrigation System - Disponível em: <<https://www.instructables.com/id/Raspberry-Pi-Controlled-Irrigation-System/>> Acesso em: Outubro de 2018.

[4] Raspberry Pi Irrigation Controller - Disponível em: <<https://www.instructables.com/id/Raspberry-Pi-Irrigation-Controller/>> Acesso em: Setembro de 2018.

[5] GRIFFES, Gregory. Raspberry Pi Irrigation Controller - Disponível em: <<https://www.hackster.io/isavewater/raspberry-pi-irrigation-controller-244fc9>> Acesso em: Outubro de 2018.

[6] Irrigation Control System Based on pi - Disponível em: <<https://www.raspberrypi.org/forums/viewtopic.php?t=153046>> Acesso em : Outubro de 2018.

[7] BARNERS, Russel. Raspberry Pi Projects Book - Disponível em: <<https://www.raspberrypi.org/magpi/>> Acesso em: Setembro de 2018.

[8] Monitorando temperatura com DHT11 e Raspberry pi - Disponível em: <<https://www.filipeflop.com/blog/temperatura-umidade-dht11-com-raspberry-pi/>> Acesso em: Dezembro de 2018.

[9] Informações de clima com Raspberry Pi Zero W - Disponível em: <<https://www.filipeflop.com/blog/clima-com-raspberry-pi-zero-w/>> Acesso em: Novembro de 2018.

[10] Informações do Clima com o PocketBeagle e OpenWeatherMap - Disponível em: <<https://www.filipeflop.com/blog/pocketbeagle-e-openweathermap/>> Acesso em: Novembro de 2018.

[11] <https://openweathermap.org/>

7. ANEXOS

```
#include "gpio_sysfs.c"
#include "gpio_sysfs.h"
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <signal.h>
#include <fcntl.h>
#include <time.h>
#include <wiringPi.h>
```

```
// Arquivo de acesso a porta serial para Raspberry Pi
#define TTY "/dev/ttyS0"
```

```
//Definição de variáveis globais
#define HIGH 1
```

```

#define LOW 0

#define pino_rele 17
#define pino_sensor 26

int leitura = 0;
float temp_sensor = 0;
float umid_sensor = 0;
char api_key_openweather[35];
char cidade[15];
char pais[15];

import requests
import json

void le_infos_clima(char *p_api_key_openweather, char
*p_cidade, char *p_pais);

void main()
{
    wiringPiSetup();
    pinMode(pino_rele, OUTPUT);
    pinMode(pino_sensor, OUTPUT);

    pthreads_t a_threads;

#demais variaveis globais
api_key_openweather =
"be6d6cf3759cbc236fa4e9a512c10bd5" #coloque aqui sua
api-key do OpenWeather
cidade = "London" #coloque aqui o nome da cidade
desejada, sem acentos para nomes com espaco, substitua o
espaco por %20
pais = "uk" #coloque aqui a sigla do pais em que a cidade
esta (para o Brasil, a sigla e br)

def le_infos_clima():
    global api_key_openweather
    global cidade
    global pais

    url_http_req =
"http://api.openweathermap.org/data/2.5/weather?q="+cida
de+" "+pais+"&appid="+api_key_openweather
    dados = requests.get(url_http_req).json()

    #informacoes desejadas do JSON
    temp_atual_kelvin = dados["main"]["temp"]
    umidade = dados["main"]["humidity"]

    temp_atual_celsius = int(float(temp_atual_kelvin) -
273.15)
    umidade_atual = float(umidade)

    return temp_atual_celsius,umidade

while(True):
    try:

```

```

#Leitura do sensor
(temp_sensor,umid_sensor) =
Adafruit_DHT.read_retry(sensor,pino_sensor)

#requisita informacoes da API e faz parser do json,
filtrando as informacoes desejadas (temperatura, umidade
e nebulosidade
(temp,umidade) = le_infos_clima()
leitura = int(leitura) + 1

temperatura_str = str(temp)+" C"
umidade_str = str(umidade)+"%"

temp_sensor_str = str(umid_sensor)+" C"
umid_sensor_str = str(temp_sensor)+"%"

#escreve as informacoes lidas e a cidade referida
print "Leitura: "+str(leitura)
print "Cidade: "+cidade+", "+pais
print "Temperatura atual: "+temperatura_str
print "Umidade relativa do ar: "+umidade_str
print ""

print "Temperatura Sensor: "+temp_sensor_str
print "Umidade Sensor: "+umid_sensor_str

if umidade < 90:
    GPIO.output(pino_rele,GPIO.HIGH)
    time.sleep(5)
    GPIO.output(pino_rele,GPIO.LOW)
else:
    GPIO.output(pino_rele,GPIO.LOW)

time.sleep(5)

except KeyboardInterrupt:
    exit(1)
}

```