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.

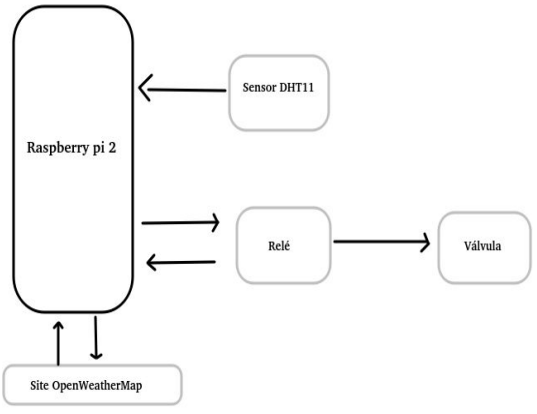

Figura 1: Diagrama de Blocos

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 DTH11, 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.

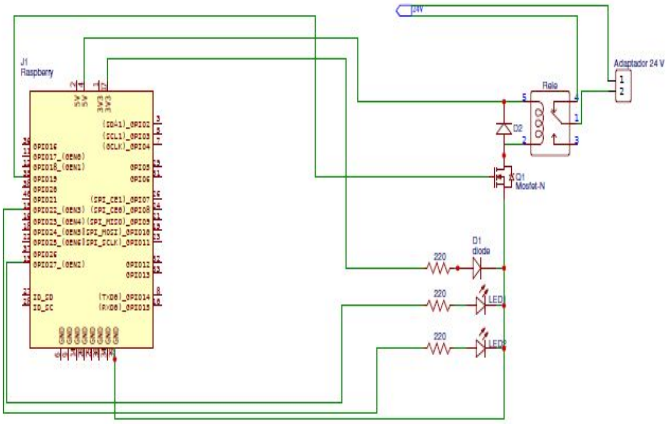

Figura 2- Projeto geral de Hardware

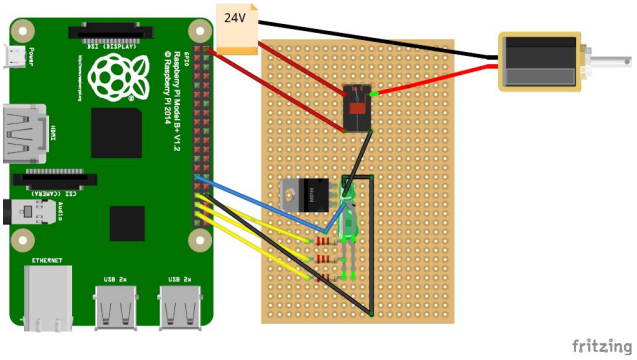

Figura 3- Esquema do circuito

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

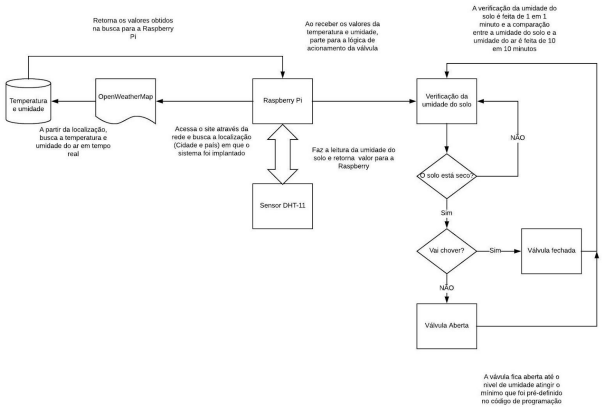

Figura 10 - Fluxograma do código

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 de 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.

No início do código temos a chamada das bibliotecas necessárias para utilizar as funções utilizadas no corpo do código. Pede-se as bibliotecas utilizadas, a inicialização das variáveis e definição dos pinos que foram utilizados. Foram diversas bibliotecas e cada uma delas tem sua funcionalidade.

A biblioteca “wiringPi.h” é a biblioteca padrão para fazer a utilização dos pinos GPIO da placa Raspberry. A biblioteca “curl.h” juntamente com a biblioteca “json.c” são as bibliotecas que permitem a conexão com o *site “OpenWeatherMap”,* fazendo a busca e o armazenamento dos dados desejados.

```
//------------------------------ ESTRUTURA DE DADOS ADQUIRIDOS DO SITE ------------------------------

clima_s le_infos_clima()
{
    char request[300]={0};
    clima_s infos;

    //Construindo url de busca das informações:
    char host_address[]= "http://api.openweathermap.org";
    char url_http_req[500] = "http://api.openweathermap.org/data/2.5/weather?q=";
    strcat(url_http_req,CIDADE);
    strcat(url_http_req,",");
    strcat(url_http_req,PAIS);
    strcat(url_http_req, "&appid=");
    strcat(url_http_req, API_KEY_OPENWEATHER);

    //Mostrando URL requisitada
    if(leitura == 0)
    {
        printf("Requisitando na URL: %s\n", url_http_req);
    }
    printf("Lendo informacoes de clima\n");

    CURL *curl;
    CURLcode res;

    curl = curl_easy_init();
    if(curl) {
        curl_easy_setopt(curl, CURLOPT_URL, url_http_req);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void*)&memoria);
        //curl_easy_setopt(curl, CURLOPT_HTTPGET, 1L);
        // Requisição GET:
        res = curl_easy_perform(curl);

        if(res != CURLE_OK)
        {
            printf("\nErro qualquer\n");
        }else
        {
            infos.temperatura_atual = extrai_temp(memoria.memory); //Recebe temperatura do site
            infos.umidade_atual = extrai_umid(memoria.memory);     //Recebe umidade do site
        }
        // Cleanup
        curl_easy_cleanup(curl);
    }

    return infos;
}
```

Figura 4 - Código final do projeto (Parte 1)

Podemos observar na Figura 4 a função que faz a conexão e a busca da temperatura e umidade através da rede. Ela faz a requisição através da biblioteca “curl.h” e retorna uma string que contém diversas informações.

Após isso, fazemos o *parser* do json através da biblioteca "json.c", onde buscamos apenas as informações que desejamos e descartamos as demais informações.

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)*.

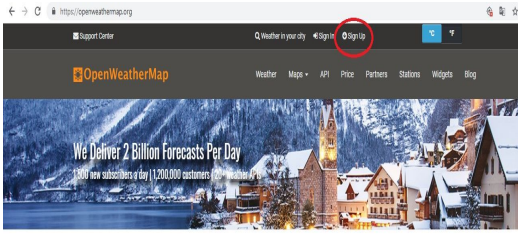

Figura 5 - Página inicial do site da *Open Weather Map*. Para fazer o cadastro basta clicar em *sign up,* como indicado em vermelho, e seguir os passos

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.

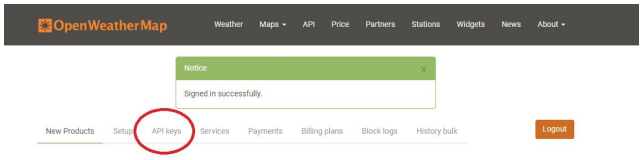

Figura 6 - Página principal, na qual o site te direciona ao término do cadastro.

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.

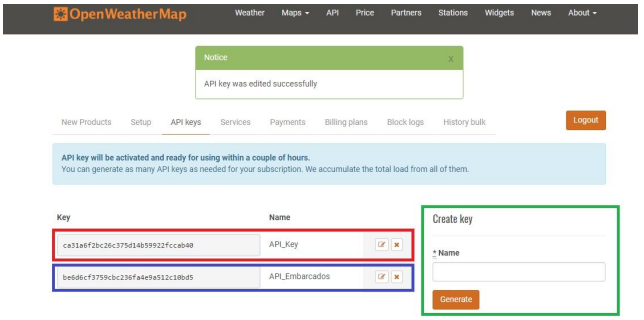

Figura 7 - Página das chaves API. Ela já vem com uma chave padrão (em vermelho) que pode ser utilizada ao terminar seu cadastro. Podem ser geradas outras chaves no campo *Create Key* (em verde). Só é necessário escolher o nome da chave e clicar em *Generate*. Após isso ela será adicionada na lista de 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 nas Figura 8 e Figura 9 a montagem final do circuito, com exceção da válvula.

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.

Como ideia para aperfeiçoamento do projeto, podemos fazer a melhoria do protótipo, implementação de um aplicativo de celular que monitora o funcionamento do sistema, uma tela para a visualização dos dados sem a necessidade que o mesmo esteja ligado em um monitor ou em uma televisão e várias outras idéias que podem ser implementadas com a utilização da Raspberry juntamente com outros microcontroladores acoplados.

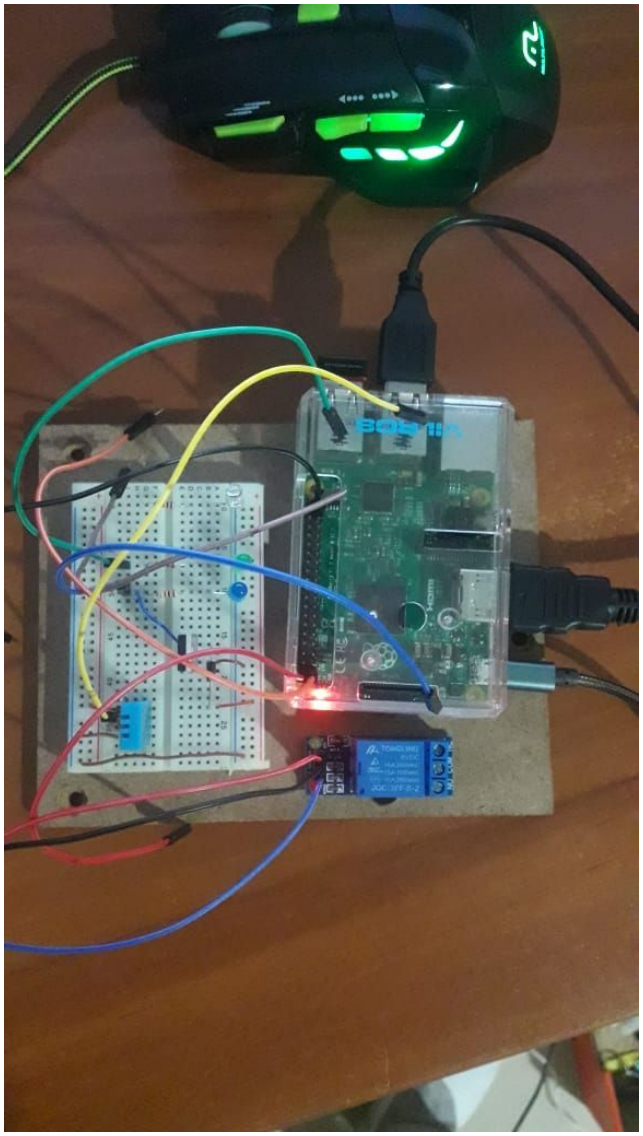

Figura 8 - Imagem do circuito montado, porém está desligado

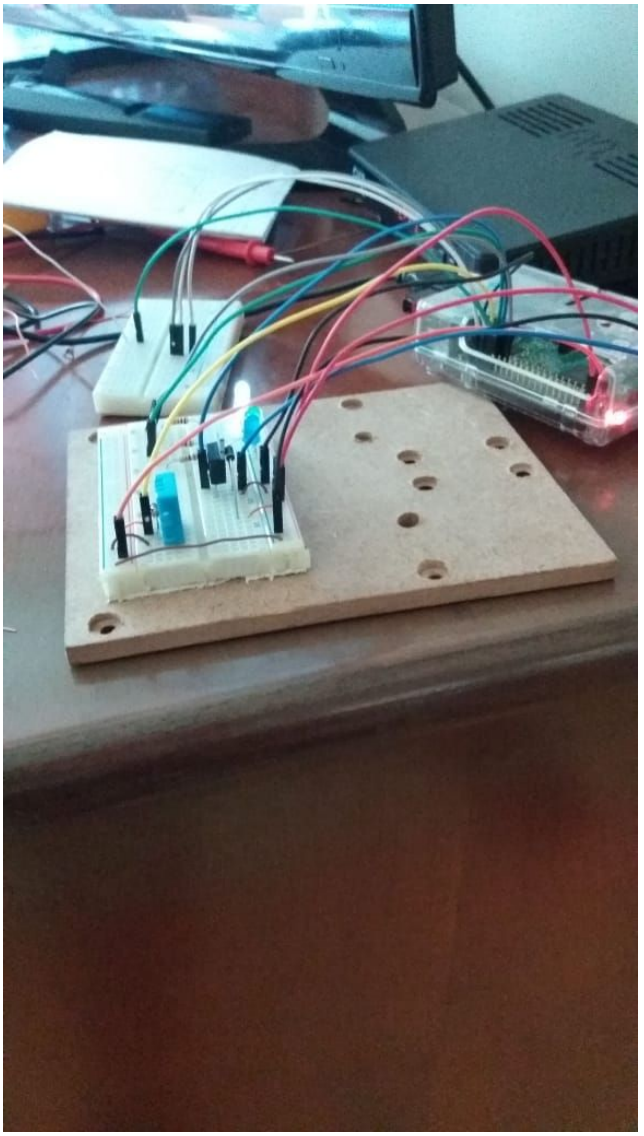

Figura 9: Circuito montado na protoboard com LED ligado, que demonstra o funcionamento do sistema

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 Nacões 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

```
/*
Código em C do projeto final de Sistemas Operacionais Embacados
Projeto: Irrigador Inteligente
Alunos:
   - Alice Fazzolino - 12/
   - Renato Cesar da Silva Agnello - 12/0053896

OBS: É necessária a instalação das bibliotecas wiringPi e cURL
O código deve ser compilado com as flags -lcurl, -lwiringPi
exemplo: gcc <nome do arquivo.c> -o <nome do arquivo> -lcurl
-lwiringPi
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wiringPi.h>
#include <stdint.h>

//Para fazer o parser dos dados obtidos pelo site
#include <json-c/json.h>

//Para requisição:
#include <curl/curl.h>

// sleep garantido tanto para windows quanto para linux/mac:
#ifdef _WIN32
   #include <windows.h>
#else
   #include <unistd.h>
#endif

//------------------------------ DEFINIÇÃO E INICIALIZAÇÃO DOS
PINOS GPIO ------------------------------
```

```
//Definição dos pinos GPIO utilizados
#define PIN_SENSOR 25   // Conectar no BCM-GPIO 26
#define LED 22                                  // Conectar no BCM-GPIO 6
#define LED_PIN_BOMBA 24     // Conectar no BCM-GPIO 19
#define PIN1_BOMBA 26
#define PIN2_BOMBA 27
#define PIN3_BOMBA 28
#define PIN4_BOMBA 29
#define PIN5_BOMBA 4
#define PIN6_BOMBA 5
#define PIN_RELE 23      // Conectar no BCM-GPIO 13

#define VALOR 60  //Valor de referência para a bomba e o relé serem
acionados

//Variável de controle para leitura do sensor
#define MAXTIMING 85

//------------------------------ VARIAVEIS GLOBAIS
------------------------------
int leitura = 0; // Contador de iterações de leitura
int segundos = 4; // Variavel que controla o tempo
int temp_sensor = 0; // Variavel que recebe a temperatura do sensor
int umid_sensor = 0; // Variavel que recebe a umidade do sensor
float h; //Umidade do sensor
float c; //Temperatura em graus Celsius do sensor

//------------------------------ STRUCT DO CLIMA ATUAL QUE
RECEBE OS DADOS VIA REDE ------------------------------

typedef struct clima_struct
{
   int temperatura_atual;
   int umidade_atual;
} clima_s;

//------------------------------ STRUCT ALOCAR MEMORIA
------------------------------

struct MemoryStruct
{
      char *memory;
      size_t size;
} memoria;

static size_t WriteMemoryCallback(void *contents, size_t size, size_t
nmemb, void *userp)
{
 size_t realsize = size * nmemb;
 struct MemoryStruct *mem = (struct MemoryStruct *)userp;
 char *ptr = realloc(mem->memory, mem->size + realsize + 1);
 if(ptr == NULL) {
   /* out of memory! */
   printf("not enough memory (realloc returned NULL)\n");
  return 0;
 }
 mem->memory = ptr;
 memcpy(&(mem->memory[mem->size]), contents, realsize);
 mem->size += realsize;
 mem->memory[mem->size] = 0;
 return realsize;
}

//------------------------------ FUNÇÃO DO SENSOR DE UMIDADE
------------------------------

int data[5] = { 0, 0, 0, 0, 0 };
void read_dht_data()
{
    uint8_t laststate   = HIGH;
    uint8_t counter                 = 0;
    uint8_t j                                       = 0, i;

    data[0] = data[1] = data[2] = data[3] = data[4] = 0;
    /* pull pin down for 18 milliseconds */
    pinMode( PIN_SENSOR, OUTPUT );
    digitalWrite( PIN_SENSOR, LOW );
    delay( 18 );

    /* prepare to read the pin */
    pinMode( PIN_SENSOR, INPUT );

    /* detect change and read data */
    for ( i = 0; i < MAXTIMING; i++ )
    {
        counter = 0;
        while ( digitalRead( PIN_SENSOR ) == laststate )
        {
            counter++;
            delayMicroseconds( 1 );
            if ( counter == 255 )
            {
                break;
            }
        }
        laststate = digitalRead( PIN_SENSOR );
        if ( counter == 255 )
            break;
        /* ignore first 3 transitions */
        if ( (i >= 4) && (i % 2 == 0) )
        {
            /* shove each bit into the storage bytes */
            data[j / 8] <<= 1;
            if ( counter > 50 )
                data[j / 8] |= 1;
            j++;
        }
    }
    /*
     * check we read 40 bits (8bit x 5 ) + verify checksum in the last byte
     * print it out if data is good
     */
    if ( (j >= 40) && (data[4] == ( (data[0] + data[1] + data[2] + data[3])
& 0xFF) ) )
    {
        h = (float)((data[0] << 8) + data[1]) / 10;

        if ( h > 100 )
        {
            h = data[0];               // for DHT11
        }
        c = (float)(((data[2] & 0x7F) << 8) + data[3]) / 10;
        if ( c > 125 )
        {
            c = data[2];               // for DHT11
        }
        if ( data[2] & 0x80 )
        {
            c = -c;
        }
    }
```

```
}

//------------------------------ CONVERTE A TEMPERATURA DA REDE PARA INT ------------------------------

int extrai_temp(char* conteudo)
{
	int TAM = strlen(conteudo);
	char aux[1000];
	char temp[5];
	int i = 0;
	int temp_int;

	strcpy(aux,conteudo);
	for(i = 0; i < TAM; i++)
	{
		if (aux[i] == 't' && aux[i+1] == 'e' && aux[i+2] == 'm' && aux[i+3] == 'p' && aux[i+4] == '"' && aux[i+5] == ':')
		{
			temp[0] = aux[i+6];
			temp[1] = aux[i+7];
			temp[2] = aux[i+8];
			sscanf(temp, "%d", &temp_int);
		}
	}
	return (temp_int-273);
}

//------------------------------ CONVERTE A UMIDADE DA REDE PARA INT ------------------------------

int extrai_umid(char* conteudo)
{
	int TAM = strlen(conteudo);
	char aux[1000];
	char umid[3];
	int i = 0;
	int umid_int;

	strcpy(aux,conteudo);

	for(i = 0; i < TAM; i++)
	{
		if (aux[i] == 'm' && aux[i+1] == 'i' && aux[i+2] == 'd' && aux[i+3] == 'i' && aux[i+4] == 't' && aux[i+5] == 'y' && aux[i+6] == '"' && aux[i+7] == ':')
		{
			umid[0] = aux[i+8];
			umid[1] = aux[i+9];
			sscanf(umid, "%d", &umid_int);
		}
	}

	return umid_int;
}
//------------------------------ CONFIGURAÇÃO DA API/CIDADE ------------------------------

char API_KEY_OPENWEATHER[] = "be6d6cf3759cbc236fa4e9a512c10bd5"; //coloque aqui sua api-key do OpenWeather
char CIDADE[] = "Gama";  //coloque aqui o nome da cidade desejada, sem acentos para nomes com espaco, substitua o espaco por %20
char PAIS[] = "BR"; //coloque aqui a sigla do pais em que a cidade esta (para o Brasil, a sigla eh br)

//------------------------------ FUNÇÃO SLEEP ------------------------------

void espera(int seconds){
  #ifdef _WIN32
    Sleep(1000 * seconds);
  #else
    sleep(seconds);
  #endif

//------------------------------ ESTRUTURA DE DADOS ADQUIRIDOS DO SITE ------------------------------

clima_s le_infos_clima(){
  char request[300]={0};
  clima_s infos;

  //Construindo url de busca das informações:
  char host_address[]= "http://api.openweathermap.org";
  char url_http_req[500] = "http://api.openweathermap.org/data/2.5/weather?q=";
  strcat(url_http_req,CIDADE);
  strcat(url_http_req,",");
  strcat(url_http_req,PAIS);
  strcat(url_http_req, "&appid=");
  strcat(url_http_req, API_KEY_OPENWEATHER);

  //Mostrando URL requisitada
  if(leitura == 0)
  {
		printf("Requisitando na URL: %s\n", url_http_req);
}

printf("Lendo informacoes de clima\n");
  CURL *curl;
  CURLcode res;

  curl = curl_easy_init();
  if(curl) {
    curl_easy_setopt(curl, CURLOPT_URL, url_http_req);
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void*)&memoria);
    //curl_easy_setopt(curl, CURLOPT_HTTPGET, 1L);
    // Requisição GET:
    res = curl_easy_perform(curl);
    if(res != CURLE_OK)
    {
			printf("\nErro qualquer\n");
		}else
		{
			infos.temperatura_atual = extrai_temp(memoria.memory); //Recebe temperatura do site
			infos.umidade_atual = extrai_umid(memoria.memory);     //Recebe umidade do site
		}
    // Cleanup
    curl_easy_cleanup(curl);
  }
  return infos;
}
```

```
//------------------------------ FUNÇÃO MAIN ------------------------------

int main(){
    curl_global_init(CURL_GLOBAL_ALL);
  printf("Iniciando o programa!\n\n");

  //Inicialização da biblioteca WiringPi e da configuração dos pinos
como OUTPUT
  wiringPiSetup();
  pinMode(LED,OUTPUT);
  pinMode(LED_PIN_BOMBA,OUTPUT);
  pinMode(PIN1_BOMBA,OUTPUT);
  pinMode(PIN2_BOMBA,OUTPUT);
  pinMode(PIN3_BOMBA,OUTPUT);
  pinMode(PIN4_BOMBA,OUTPUT);
  pinMode(PIN5_BOMBA,OUTPUT);
  pinMode(PIN6_BOMBA,OUTPUT);
  pinMode(PIN_RELE,OUTPUT);
  digitalWrite(LED,LOW);
  clima_s clima;

  //Loop infinito para aquisição e comparação dos dados obtidos
  while(1)
  {
        do
        {
              read_dht_data();
        }while(c == 0.0 && h == 0);
        espera(4);

        //Condicional para o funcionamento do projeto
        if(h >= VALOR)
        {
              if(clima.umidade_atual <= 50)
              {
                    printf("UMIDADE ALTA! VALOR:
%.1f\nATIVANDO BOMBA!\n", h);
                    printf("RELE ABERTO ATE A
UMIDADE TER VALOR MENOR QUE %d!\n", VALOR);
                    while(h >= VALOR)
                    {
                          read_dht_data();

digitalWrite(PIN_RELE,HIGH);
                          digitalWrite(LED, LOW);

digitalWrite(LED_PIN_BOMBA,HIGH);

digitalWrite(PIN1_BOMBA,HIGH);

digitalWrite(PIN2_BOMBA,HIGH);

digitalWrite(PIN3_BOMBA,HIGH);

digitalWrite(PIN4_BOMBA,HIGH);

digitalWrite(PIN5_BOMBA,HIGH);

digitalWrite(PIN6_BOMBA,HIGH);
                          printf("Umidade sensor:
%.1f\n", h);
                          espera(1);
                    }
                    printf("FECHANDO
VALVULA!\n\n");
digitalWrite(LED_PIN_BOMBA,LOW);
                          digitalWrite(PIN1_BOMBA,LOW);
                          digitalWrite(PIN2_BOMBA,LOW);
                          digitalWrite(PIN3_BOMBA,LOW);
                          digitalWrite(PIN4_BOMBA,LOW);
                          digitalWrite(PIN5_BOMBA,LOW);
                          digitalWrite(PIN6_BOMBA,LOW);
                          digitalWrite(PIN_RELE,LOW);
              }
        }
        else
        {
              digitalWrite(LED,HIGH);
              digitalWrite(LED_PIN_BOMBA,LOW);
              digitalWrite(PIN1_BOMBA,LOW);
              digitalWrite(PIN2_BOMBA,LOW);
              digitalWrite(PIN3_BOMBA,LOW);
              digitalWrite(PIN4_BOMBA,LOW);
              digitalWrite(PIN5_BOMBA,LOW);
              digitalWrite(PIN6_BOMBA,LOW);
              digitalWrite(PIN_RELE,LOW);
        }
memoria.memory = malloc(1);
        memoria.size = 0;
   //requisita informacoes da API e faz parser do json, filtrando as
informacoes desejadas (temperatura e umidade)
    clima = le_infos_clima();
    leitura++;
    printf("\n---------- Leitura: %d ----------\n\n", leitura);
    printf("Temperatura na cidade %s, %s: %d °C\n",CIDADE, PAIS,
clima.temperatura_atual);
    printf("Umidade na cidade %s, %s: %d %%\n",CIDADE, PAIS,
clima.umidade_atual);
    printf("Temperatura Sensor: %.f °C\n", c);
    printf("Umidade Sensor: %.f %%\n\n", h);

    free(memoria.memory);
   curl_global_cleanup();
  }
}
```