PONTO DE CONTROLE 2 - 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. JUSTIFICATIVA

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 importante ressaltar que o protótipo será feito pensando em vasos e pequenos jardins, mas é totalmente possível, com algumas modificações, utilizar o projeto em pequenas e grandes plantações .

2. OBJETIVOS

Este projeto tem como principal objetivo desenvolver um sistema de irrigação automatizado e em tempo real que irrigará jardins ou plantações de forma rápida e econômica.

O objetivo é que o controle de irrigação seja realizado através da raspberry que se conectará à internet e realizará uma rápida pesquisa no clima da região, detectando assim se é necessário efetuar a irrigação e por quanto tempo realizar a mesma.

Assim como outros objetivos:

1. Estudar e analisar as formas de irrigação;
2. Estudar o sistema linux;
3. Estudar a Raspberry pi;
4. Testar e verificar o funcionamento do protótipo.

3. REQUISITOS

- **Requisitos Mínimos:**
    - Integração da Raspberry com os demais componentes do projeto;
    - Controle da vazão de água e do tempo de irrigação;
    - Controle da umidade do solo situado na área de atuação do dispositivo.
    - Conexão entre a placa e a rede, para verificação da previsão do tempo (pensando em melhorar o consumo/desperdício de água);

4. BENEFÍCIOS

Além de manter o jardim vivo, bonito e facilitar a vida do usuário que não precisaria mais inspecionar o sistema quando estivesse em execução (para achar algum problema), também haveria uma significante economia de água no local, devido ao fato que ter conhecimento sobre o clima da região fará com que o sistema esteja ciente de uma provável chuva e com isso irá cancelar a irrigação daquele dia.

5. DESENVOLVIMENTO

5.1 DESCRIÇÃO DO HARDWARE

5.1.1 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.

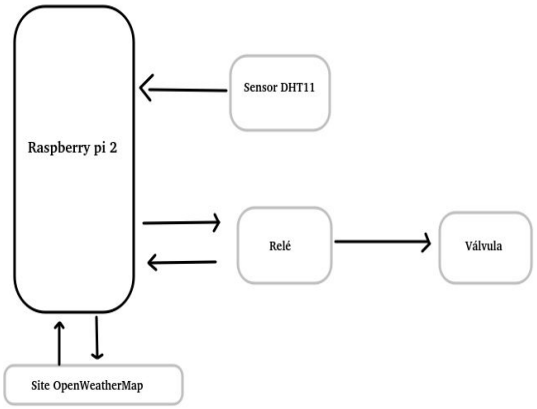

Figura 1: Diagrama de Blocos

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.

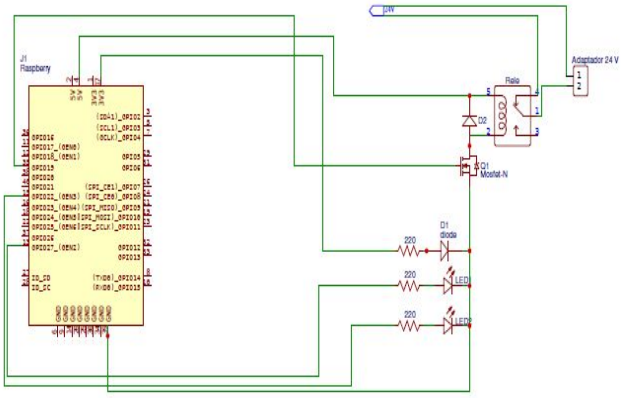

Figura 2- Projeto geral de Hardware

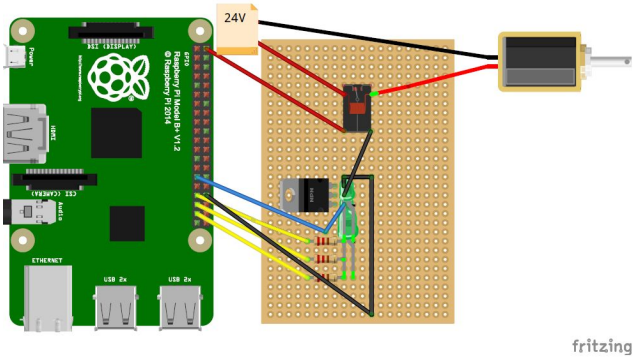

Figura 3- Esquema do circuito

5.2 DESCRIÇÃO DE SOFTWARE

5.2.1 FLUXOGRAMA

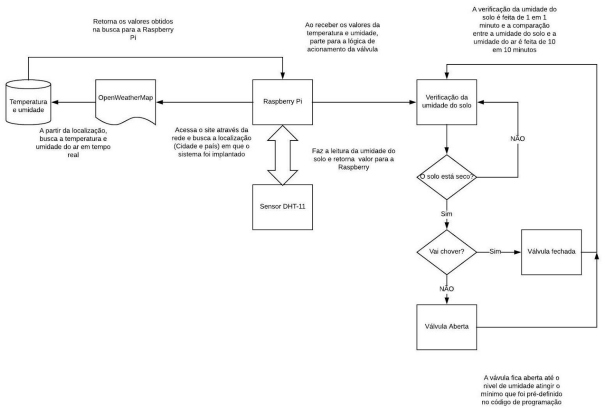

Figura 10 - Fluxograma do código

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.

5.2.2 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:

```c
#include "gpio_sysfs.c"
#include "gpio_sysfc.h"
#include <stdio.h>
#include <stdlib.h>
#include <pthreads.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_openweather[35];
char cidade[15];
char pais[15];
```

Figura 11 - Inicialização das bibliotecas e variáveis globai

```c
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="+cidade+","+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
```

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

```c
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)
```

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

6. REVISÃO BIBLIOGRÁFICA

[1] Ministério do Meio Ambiente - Água. Disponível em:< http://www.mma.gov.br>. Acesso em: Março de 2019.

[2] Organização das Nacões Unidas no Brasil - A ONU e a Água - Disponível em:

<https://nacoesunidas.org/acao/agua/> Acesso em: Março de 2019.

[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. ANEXOS

```
#include "gpio_sysfs.c"
#include "gpio_sysfc.h"
#include <stdio.h>
#include <stdlib.h>
#include <pthreads.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_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)
}
```