FACULDADE DE TECNOLOGIA – FATEC SANTO ANDRÉ

Caio Felipe dos Santos

Vítor Amaducci Negocia

Wesley da Silva Viana

CONTROLE UMIDIFICADOR DE AR

Santo André

2019

Caio Felipe dos Santos
Vítor Amaducci Negocia
Wesley da Silva Viana

CONTROLE UMIDIFICADOR DE AR

Santo André
2019

SUMÁRIO

1. **OBJETIVO** 4
2. **CIRCUITO** 5
3. **FLUXOGRAMA** 6
4. **PROGRAMAÇÃO** 7
5. **CONCLUSÃO** 10
6. **BIBLIOGRAFIA** 11

1. OBJETIVO

Este trabalho visa o controle da umidade do ambiente de acordo com a vontade do usuário.

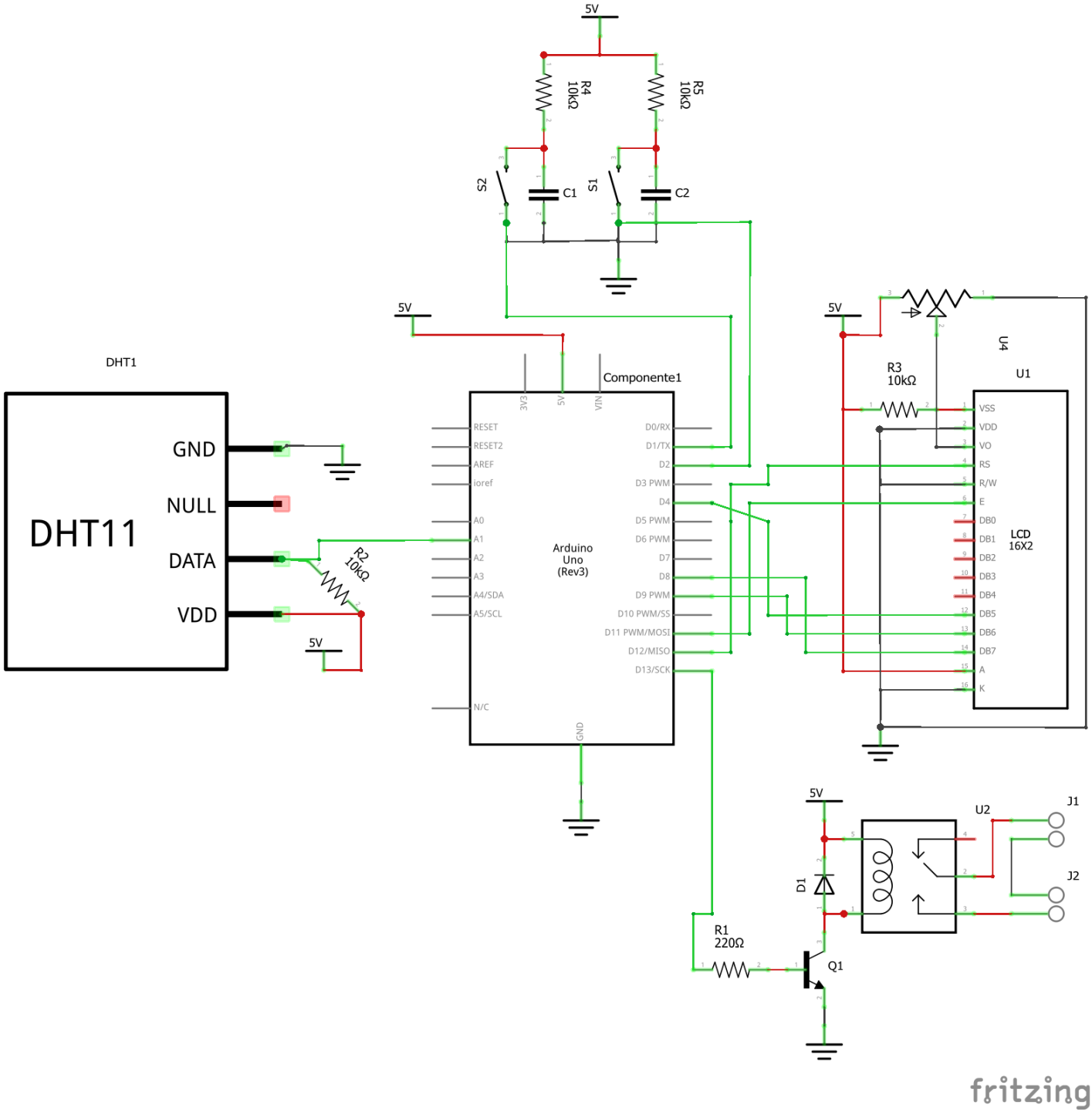
Para isso foi utilizado um sensor DHT11, ele permite fazer a medição de temperatura e umidade relativa do ar, podendo coletar dados entre 0 a 50ºC e a umidade entre 20 a 90%, o sensor teve como processador de dados o microcontrolador atmega. O elemento utilizado neste sensor é o termistor tipo NTC, essa parte é utilizada para medir temperatura, já para medir umidade utiliza outro sensor embutido que é o HR202, então ambos os sensores conversam via serial.

Com a informação do sensor tivemos a base para comparar as informações do usuário, sendo essas informações vindas de dois botões, esses botões são do tipo pull up eles informam qual a umidade desejada, sendo um para aumentar a umidade e outro para diminuir.

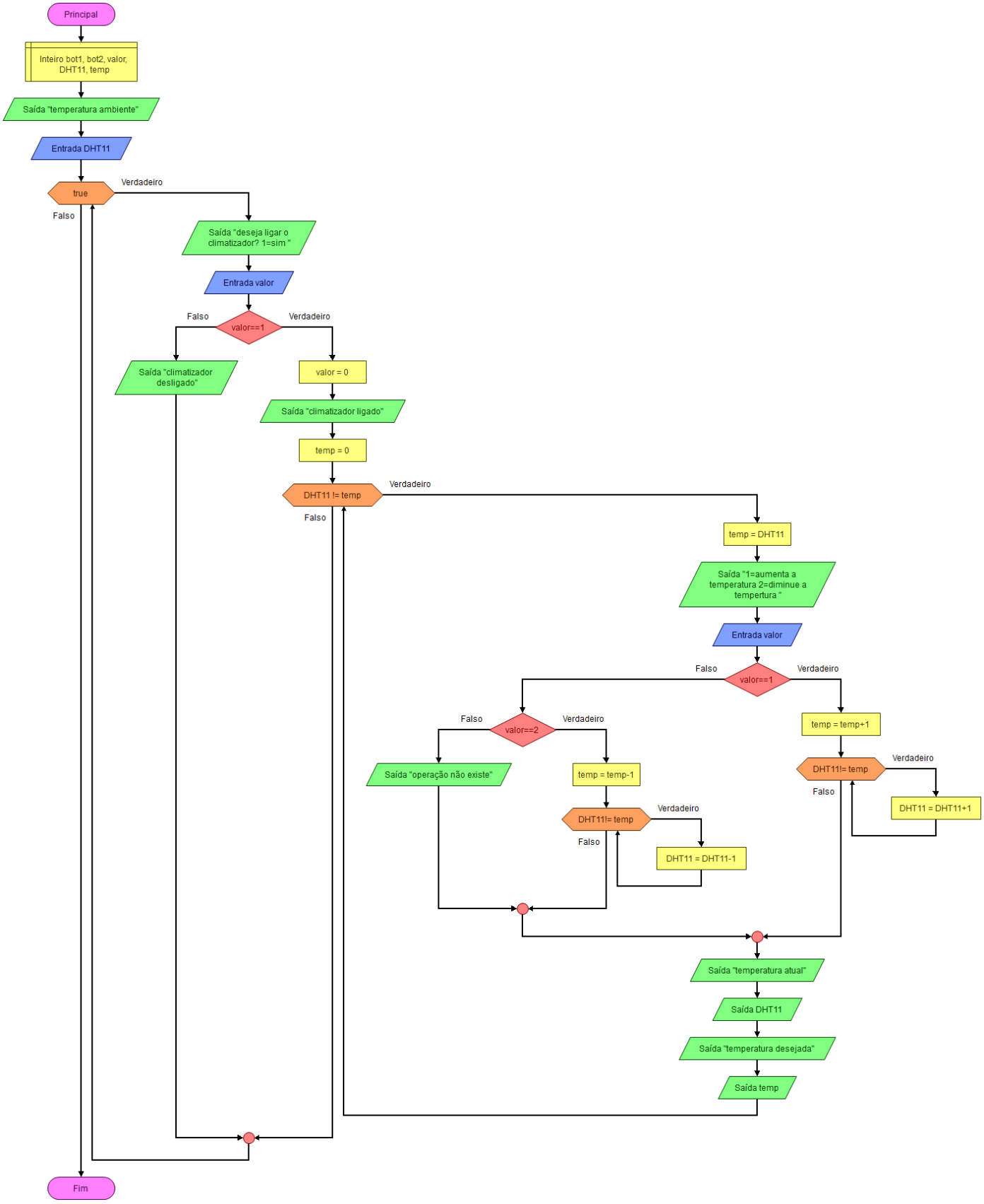
Para acionar a carga precisamos de um circuito isolador para isso utilizamos um modulo relé, nele através do chaveamento um transistor controlado pelo microcontrolador atmega é possível acionar um relé, deste modo terá uma isolação galvânica entre o circuito de controle e o circuito de potência.

E todas as informações são passadas por um lcd 16x2, tendo em vista que na primeira linha ele mostra a umidade em tempo real, já na segunda linha é posto a umidade almejada.

2. CIRCUITO


5V
R4
10kΩ
R5
10kΩ
S2
C1
S1
C2
DHT1
DHT11
GND
NULL
DATA
VDD
R2
10kΩ
Componente1
Arduino
Uno
(Rev3)
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
R3
10kΩ
U4
U1
LCD
16X2
VSS
VDD
VO
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
A
K
U2
J1
J2
D1
R1
220Ω
Q1
fritzing

3. FLUXOGRAMA



4. PROGRAMAÇÃO

ALTA LINGUAGEM

```
#include <LiquidCrystal.h>
#include "DHT.h"
#define DHTPIN A1
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 9, d7 = 8;
LiquidCrystal  lcd(rs, en, d4, d5, d6, d7);
int BOT1 = 2;
int BOT2 = 3;
volatile float UMID_DES = 0;
int CLIMA = 13;
void setup()
{
  pinMode(BOT1, INPUT);
  pinMode(BOT2, INPUT);
  pinMode(CLIMA, OUTPUT);
  lcd.begin(16, 2);
  dht.begin();
  attachInterrupt(digitalPinToInterrupt(BOT1),aumenta_umidade, FALLING);
  attachInterrupt(digitalPinToInterrupt(BOT2),diminui_umidade, FALLING);
}

void loop()
{
  float h = dht.readHumidity();

    lcd.setCursor(0, 0);
    lcd.print("Umidade: ");
    lcd.print(h);
    lcd.setCursor(0, 1);
    lcd.print("Desejada ");
    lcd.print(UMID_DES);
    lcd.print("%");

  float UMID = dht.readHumidity(); {
    if (h < UMID_DES)
    {
      digitalWrite(CLIMA, HIGH);
    }
    else
    {
      digitalWrite(CLIMA, LOW);
    }
  }
}
void aumenta_umidade()
{
  UMID_DES = UMID_DES + 1;
}

void diminui_umidade()
{
  UMID_DES = UMID_DES - 1;
}
```

BAIXA LINGUAGEM

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <minhaCom.h>
#include <minhasMacros.h>
#include <LiquidCrystal.h>
#include "DHT.h"
#define DHTPIN DDC1
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
const int rs = DDB4, en = DDB3, d4 = DDD4, d5 = DDD5, d6 = DDB1, d7 = DDB0;
LiquidCrystal  lcd(rs, en, d4, d5, d6, d7);
int BOT1 = DDD2;
int BOT2 = DDD3;
volatile float UMID_DES = 0;
int CLIMA = DDB5;

ISR(INT0_vect)
{
UMID_DES = UMID_DES + 1;
}
ISR(INT1_vect)
{
UMID_DES = UMID_DES - 1;
}
void iniInterrupt1(void){
        EIMSK |=(1 << INT0);
        EICRA |=(1 << ISC00);
        sei();
void iniInterrupt2(void){
        EIMSK |=(1 << INT1);
        EICRA |=(1 << ISC00);
        sei();
}
void setup()
{
        initUSART();
        //configura pb2 como entrada e liga seu pullup
        clearBit(DDRD, DDD2);
        setBit(DDRD, DDD2);
        clearBit(DDRD, DDD3);
        setBit(DDRD, DDD3);
        setBit(DDRB, DDB5);
        lcd.begin(16, 2);
        dht.begin();
        initInterrupt();
}
void loop()
{
    float UMID = dht.readHumidity();
    lcd.setCursor(0, 0);
    lcd.print("Umidade: ");
    lcd.print(UMID);
    lcd.setCursor(0, 1);
    lcd.print("Desejada ");
    lcd.print(UMID_DES);
    lcd.print("%");
    if (UMID < UMID_DES)
```

```
    {
      setBit(PORTB, PB5);
    }
    else
    {
      clearBit(PORTB, PB5);
    }
  }
}
```

5. CONCLUSÃO

Concluímos com o desenvolvimento deste trabalho, que a linguagem de baixo nível (em C++) é melhor para o microprocessador da ATMEGA, pois ao mover bits de registradores o compilador precisa converter menos informação, a tal modo que a programação tenha tamanhos pequenos.

Entendemos o funcionamento do sensor DHT11, lembrando que o mesmo tem o “conversor ADC” embutido, assim ele envia sinais binários de umidade e temperatura para o processador.

Trabalhamos com as interrupções, então ao criar uma rotina de trabalho a interrupção é para operar entre o desenvolver das tarefas.

Por fim podemos ver o que com a programação embarcada temos menores espaço utilizado de armazenamento e mais rápida é a resposta processada.

6. BIBLIOGRAFIA

DATASHEET DHT11. Acesso em: https://bit.ly/2vfZGmX

DATASHEET ARDUINO. Acesso em: https://bit.ly/2DwD6e6

DATASHEET DISPLAY. Acesso em: https://bit.ly/2XxxQi6