Universidade Federal de Uberlândia – UFU
Faculdade de Engenharia Mecânica – FEMEC
Graduação em Engenharia Mecatrônica
Sistemas Digitais para Mecatrônica

Prof. Eder Alves Moura

TRABALHO 2
SISTEMAS DIGITAIS PARA MECATRÔNICA

Internet Of Things - IOT

Jean Robert da Cunha Marquez - 11621EMT008

João Vitor Barbosa Rocha - 11711EMT007

Lucas Eduardo Marra de Lima - 11611EMT002

Paulo Vítor Arrais de Lima - 11421EMT024

Rafael Marques Borges Pacheco - 11721EMT019

Uberlândia, agosto de 2022

Sumário

1. INTRODUÇÃO

Nos últimos anos, a IoT (Internet das Coisas) se tornou uma das tecnologias mais importantes do século XXI. Agora que podemos conectar objetos do cotidiano - eletrodomésticos, carros, termostatos, babás eletrônicas - à Internet por meio de dispositivos incorporados, é possível uma comunicação perfeita entre pessoas, processos e outras coisas.

Por meio da computação de baixo custo, nuvem, big data, análise avançada e tecnologias móveis, coisas físicas podem compartilhar e coletar dados com o mínimo de intervenção humana. Nesse mundo hiperconectado, os sistemas digitais podem gravar, monitorar e ajustar cada interação entre itens conectados. O mundo físico encontra o mundo digital, e eles trabalham em conjunto.

Este trabalho consiste no desenvolvimento de um projeto de IOT capaz de simular uma aplicação real em uma casa inteligente. A situação escolhida utiliza o sensor DHT11, que trabalha obtendo informações a respeito da umidade e temperatura a serem repassadas ao servidor. O usuário acessa uma interface contendo essas informações, e decide por ligar ou desligar um ventilador, sendo possível realizar esta operação com apenas um clique. O sensor e o relé responsável pelo acionamento do atuador, estão conectados a um Raspberry Pi, que envia e recebe informações do servidor nele hospedado.

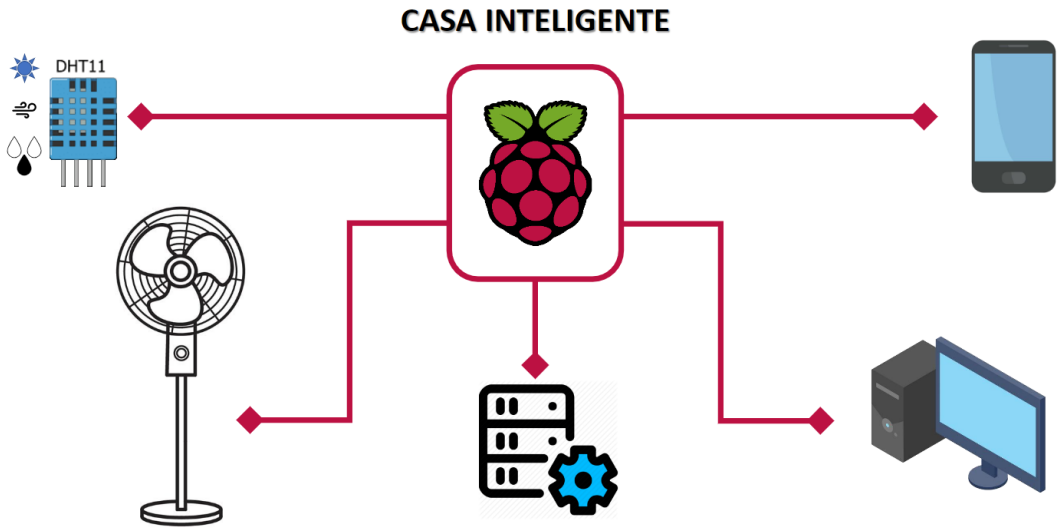

Figura 1: Implementação

2. MATERIAIS UTILIZADOS

Os materiais abaixo foram escolhidos por sua fácil manipulação, além de integrantes do grupo possuírem as partes desejadas. Sendo assim, a união dos microcontroladores se tornou natural, fazendo com que o trabalho seja mais completo e englobe necessidades cotidianas, como o uso de várias ferramentas que estão ao alcance.

Sendo assim, abaixo estão os materiais eletrônicos utilizados:

- Raspberry Pi 4
- Arduino Uno R3
- Módulo sensor de temperatura e umidade DHT11
- Relé 5v (250VAC @ 7A ou 125VAC @ 10A)
- Jumpers para as conexões
- Ventilador doméstico
- Tomadas

Figura 2: Raspberry Pi 4

Figura 3: Arduino Uno R3

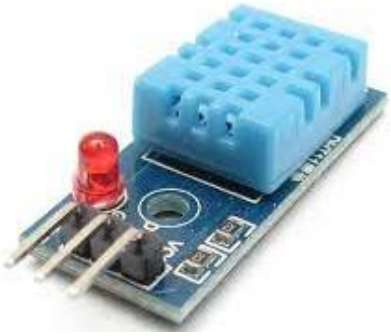

Figura 4: Módulo sensor DHT11

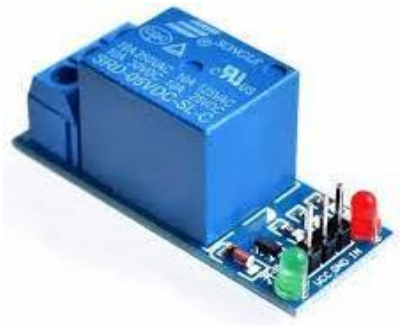

Figura 5: Relé 5 V

3. DESENVOLVIMENTO

3.1. INSTALAÇÃO DO FLASK

Antes de mais nada, é interessante discutir e apresentar os conceitos por trás do Flask. Flask é um ambiente de trabalho simples, escrito em python e de código aberto (open source ou licença BSD). Sua função é ser uma plataforma que permite desenvolvimento web de maneira descomplicada. Abordando um pouco de história, foi criado em 2010 por Armin Ronacher.

É caracterizado por sua simplicidade, como será visto no código em python apresentado em anexo, mas mesmo assim se torna necessário a existência de arquivos para alimentar o site, algo que é intrínseco do desenvolvimento web. No caso do Flask, esses arquivos estão em menor quantidade.

Também se destaca pela rapidez de desenvolvimento, uma que trabalhando com este frame-work, o desenvolvedor está focado somente no que interessa para a aplicação ser eficiente e robusta. A robustez de seus projetos vem do fato de que sua arquitetura é muito personalizável, fazendo com que o grupo que desenvolve não se limita a aspectos rígidos da aplicação.

Assim, neste projeto toda a comunicação será feita com um servidor web, e para criar o servidor usando o flask, é necessário instalá-lo no Raspberry Pi. O primeiro passo é executar o comando abaixo para instalar o pip. O pip é usado para instalar diferentes pacotes para Python.

```
sudo apt-get install python-pip
```

A seguir, deve-se usar o pip para instalar o flask no raspberry pi executando o comando abaixo:

```
sudo pip install flask
```

Uma vez que o código acima é executado com sucesso, significa que o Flask está instalado.

O próximo passo, foi verificar no navegador. Inicialmente, é necessário conhecer o endereço IP do Raspberry Pi. Para descobrir o IP, basta usar o comando “ifconfig” no terminal do Raspberry, e ele mostrará o seu endereço IP. Tendo em mãos o IP do Raspberry Pi, o mesmo foi inserido no navegador do computador. É possível usar o endereço IP em qualquer dispositivo para acessar o servidor web. Para tal, é necessário apenas certificar-se de que todos os dispositivos estejam conectados à mesma rede.

3.2. MONTAGEM

A montagem do sistema utilizado é bem simples. A Fig. XX ilustra bem como os componentes estão dispostos. Primeiramente, o Arduino e o Raspberry Pi estão se comunicando via serial (uma comunicação comum e confiável). Aquele é responsável pela aquisição dos dados do sensor DHT11, ligado no Pino Digital 8. Já o Raspberry Pi 4, que possui conexão Web por meio do Flask, é responsável por acionar o relé, que no caso deste trabalho seria ligar e desligar a ventilação da sala.

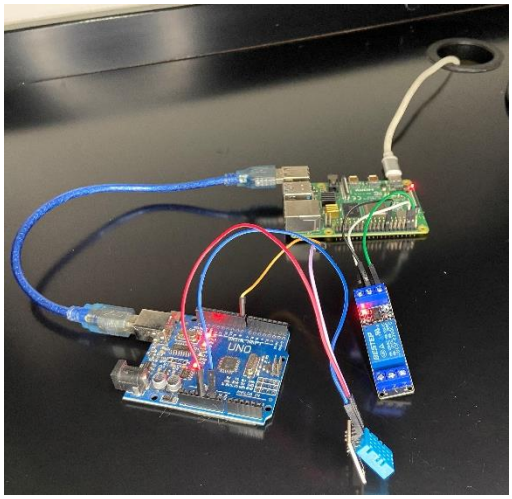

Figura 6: Montagem dos componentes

É interessante observar a distribuição dos pinos e suas respesctivas funções, para isso, dispõe-se as figuras a seguir.

PIN	NAME	NAME	PIN
01	3.3V DC Power	5V DC Power	02
03	GPIO02 (SDA1,I²C)	5V DC Power	04
05	GPIO03 (SDL1,I²C)	Ground	06
07	GPIO04 (GPCLK0)	GPIO14 (TXD0, UART)	08
09	Ground	GPIO15 (RXD0, UART)	10
11	GPIO17	GPIO18(PWM0)	12
13	GPIO27	Ground	14
15	GPIO22	GPIO23	16
17	3.3V DC Power	GPIO24	18
19	GPIO10 (SP10_MOSI)	Ground	20
21	GPIO09 (SP10_MISO)	GPIO25	22
23	GPIO11 (SP10_CLK)	GPIO08 (SPI0_CE0_N)	24
25	Ground	GPIO07 (SPI0_CE1_N)	26
27	GPIO00 (SDA0, I²C)	GPIO07 (SCL0, I²C)	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12 (PWM0)	32
33	GPIO13 (PWM1)	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Figura 7: Pinagem Raspberry PI 4

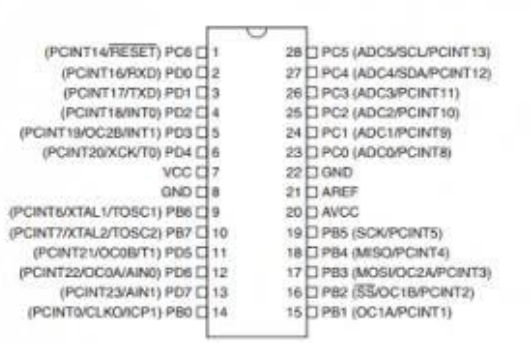

Figura 8: Pinagem Arduino Uno

3.3. FUNCIONAMENTO

O projeto foi realizado unindo uma gama de linguagens e conceitos importantes para a área de Sistemas Embarcados. Primeiramente, a parte de Front-End foi realizada com HTML e CSS, a primeira possui a função de linkar o Back-End, feito em python, com a página Web. Mais precisamente, sua função é permitir que o navegador leia e envie informações de outras mídias para uma página na web. Já o código em CSS é chamado de estático, pois se limita a uma estilização da página como um todo. Sendo assim, caso seja necessário mudar algo visual no site produzido pelo HTML, é possível alterar somente o CSS, que é chamado pelo primeiro. A Fig. Mostra um recorte da tela criada para interação entre o site e o conjunto montado.

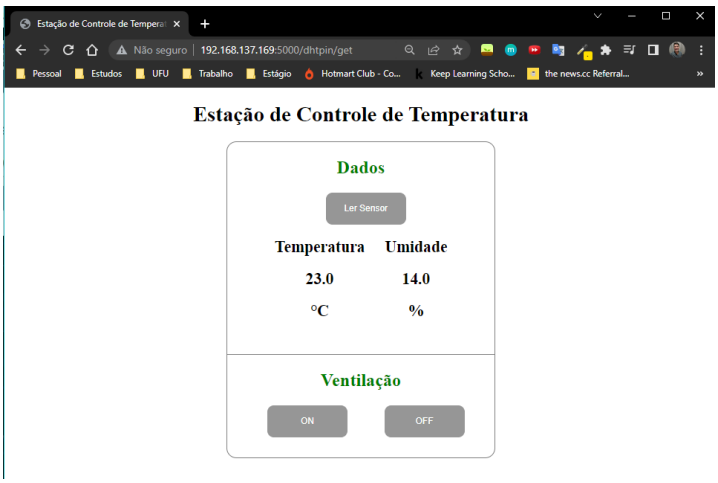

Figura 9: Página WEB criada com HTML + CSS

Partindo agora para o Back-End, utilizou-se da linguagem Python 3 para interação entre o Raspberry, HTML e CSS responsáveis pela criação da página acima detalhada. Vale ressaltar que também foi necessária a programação na linguagem C++, pela IDE do Arduino. Sendo esta necessária para coletar as informações do Sensor DHT11 e enviar ao Raspberry por meio da comunicação serial já ilustrada.

Em suma, o Raspberry (por meio do python) é responsável por repassar os dados coletados pelo Arduino para a página Web, quando o usuário apertar o botão “Ler Sensor”. Ao observar os parâmetros do ambiente, o usuário pode escolher entre ligar e desligar a ventilação. Assim, ao precisar o botão “ON”, o site envia tal informação ao Raspberry Pi que liga o relé, caso esteja desligado, proporcionando assim alimentação ao Ventilador.

Vale ressaltar que os códigos citados estão todos em anexo. Também está disposto no YouTube um vídeo com os participantes explicando cada aspecto do projeto, acessível pelo Link:

https://youtu.be/1xGqbHpsXKE

4. CONCLUSÃO

Conforme já apresentado, o usuário pode obter informações sobre as condições de temperatura e umidade com um simples acesso ao servidor. Além de receber informações, ele pode controlar um ventilador à distância. Os resultados satisfatórios corroboram a simplicidade e aplicabilidade da IoT no cotidiano, o que pode trazer conforto e segurança ao ser humano.

Apesar de a solução apresentada ser focada em uma situação específica, diversos projetos podem ser desenvolvidos utilizando o mesmo conceito. Alguns exemplos são: acionar lâmpadas, obter um status da quantidade de gás disponível no botijão da cozinha a fim de não ser pego de surpresa, nível de água em um reservatório e etc.

Por fim, ressalta-se a multidisciplinaridade deste Projeto, uma vez que foi necessário unir conhecimentos de programação propriamente dita, redes de computadores, eletrônica em geral e um pouco de criatividade. Este último sendo essencial para o cotidiano de um profissional de engenharia. Também se trata de uma oportunidade ímpar que foi aproveitada com sucesso, onde o grupo colocou um objetivo que foi atingido, sendo este controlar a temperatura e umidade de uma sala a certa distância.

5. REFERÊNCIAS BIBLIOGRÁFICAS

[1] IOT STARTERS, 2021. Disponível em: <https://www.iotstarters.com/raspberry-pi-flask-web-server-with-dht11/>. Acesso em: 10 de agosto de 2022.

[2] ORACLE. Disponível em: <https://www.oracle.com/br/internet-of-things/what-is-iot/>. Acesso em: 13 de agosto de 2022.

6. ANEXOS

- **Código HTML:**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>

<head>
<!--Chama scripts base para html e chama o arquivo .css-->
   <title>Estação de Controle de Temperatura</title>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>
   <link rel= "stylesheet" type= "text/css" href= "{{ url_for('static',filename='styles/style.css') }}">
</head>

<body>
   <h1>Estação de Controle de Temperatura</h1>
<!--Cria classe principal-->
   <div class="principal">
      <div class="column">
         <section class="container"><!--Cria subdivisões para organizar os dados-->
            <div class="item-titulo"><h1>Dados</h1></div>
            <div class="aquire">
               <a href="/dhtpin/get"> <!--Cria botão que chama ref para ler sen-->
                  <button class="botao2">
                     <span class="text-aquire">Ler Sensor</span>
                  </button>
               </a>
            </div>
            <div class="row">
               <!--cria espaço onde vai ser mostrado os dados vindos do python-->
               <div class="item-temperatura">
                  <h2>Temperatura</h2>
                  <h1 id="temperatura">{{ temperature }}</h1>
                  <h2>°C</h2>
               </div>

               <div class="vertical-separator"></div>

               <div class="item-umidade">
                  <h2>Umidade</h2>
                  <h1 id="umidade">{{ humidity }}</h1>
                  <h2>%</h2>
```

```
                </div>

                    </div>
            <!--Cria spacos e divisões para separar dados-->
            <div class="space"></div>
            <div class="separator"></div>
            <div class="item-titulo"><h1>Ventilação</h1></div>
            <div class="row">
<!--Cria botões para ligar e desligar relê de ventilação-->
              <div class='divBotao'>
                <a href="/rele/on">
                  <button id="on" class='bnn' >
                    ON
                  </button>
                </a>
              </div>

              <div class='divBotao'>
                <a href="/rele/off">
                  <button id="off" class='bnn' >
                    OFF
                  </button>
                </a>
              </div>

            </div>
            <div class="space"></div>
          </section>
        </div>
      </div>

      <script>
      </script>
</body>
</html>
```

- **Código em CSS:**

```
.botao {
  background-color: rgb(150, 150, 150);
  border-radius: 10px;
  width: 200px;
  color: #fff;
  padding: 20px;
  margin: 0px 20px 0 35px;
}

.botao2 {
  background-color: rgb(150, 150, 150);
  border-radius: 10px;
  border: solid rgb(150, 150, 150) .5px;
  width: 120px;
  color: #fff;
  padding: 15px;
  margin: 0px 20px 0 35px;
}

.ventilator {
  margin: 0px 0 0 25px;
}

.botao:hover {
  font-weight: bold;
  /* color: green; */
}

.botao2:hover {
  background-color: rgb(70, 70, 255);
  border: solid rgb(70, 70, 255) .5px;
  box-shadow: 0 0 30px 0 rgb(70, 70, 255),
     0 0 20px 0 rgb(70, 70, 255),
     0 0 10px 0 rgb(70, 70, 255) inset;
}

#on:hover {
  background-color: green;
  border: solid green .5px;
  box-shadow: 0 0 20px 0 green,
     0 0 20px 0 green,
     0 0 10px 0 green inset;
```

```
}

#off:hover {
  background-color: red;
  border: solid red .5px;
  box-shadow: 0 0 20px 0 red,
     0 0 20px 0 red,
     0 0 10px 0 red inset;
}

.bnn {
  background-color: rgb(150, 150, 150);
  border-radius: 10px;
  border: solid rgb(150, 150, 150) .5px;
  width: 120px;
  color: #fff;
  padding: 15px;
  margin: 0px 20px 0 35px;
}


.buttonON {
  background-color: green;
  border: solid green;
  box-shadow: 0 0 30px 0 green,
     0 0 30px 0 green,
     0 0 10px 0 green inset;
}


.principal {
  display: flex;
  justify-content: center;
  align-items: center;
}

body {
  display: block;
  text-align: center;

}

.space {
```

```
  width: 400px;
  height: 30px;
}

.row {
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
}

.separator {
  width: 400px;
  height: 1px;
  background-color: gray;
}

.temperatura {
  margin: 20px;
  align-self: center;
  font-size: xx-large;

}


.column {
  border: .5px solid gray;
  border-radius: 15px;
  width: 400px;
}

.item-titulo {
  color: green;
  font-size: large;
}

.vertical-separator {
  width: 30px;
}
```

- **Código C++ (Arduino):**

```
#include "DHT.h"
#define DHTPIN 8     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11

// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.

  float h = dht.readHumidity();
  float t = dht.readTemperature();   // Read temperature as Celsius (the default)

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  Serial.print(h);
  Serial.print("e");
  Serial.print(t);
  Serial.println("");
  delay(100);
}
```

- **Código em Python:**

```
# Importar bibliotecas
from flask import Flask, render_template
import RPi.GPIO as GPIO
import time
import serial

# Inicia comunicação serial arduino
comunicacaoSerial = serial.Serial('/dev/ttyUSB0', 9600)

app = Flask(__name__)

GPIO.setmode(GPIO.BOARD)
rele = 16
GPIO.setup(rele, GPIO.OUT)
GPIO.output(rele, GPIO.HIGH)


# Cria Rota principal e chama arquivo html
@app.route("/")
def main():
   return render_template('index.html')


# Cria rota dos botões
@app.route("/<pin>/<action>")
def sensor(pin, action):
   temperature = ''
   humidity = ''

   # verifica se o botão da ventilção foi acionado para ligar ou desligar
   if pin == "rele" and action == "on":
      GPIO.output(rele, GPIO.LOW)
      print('on')
   if pin == "rele" and action == "off":
      GPIO.output(rele, GPIO.HIGH)
      print('off')

   # Limpa flush da comunicação serial
   comunicacaoSerial.flush()
   comunicacaoSerial.flushInput()
   comunicacaoSerial.flushOutput()
   time.sleep(2)  # aguarda 200 ms para receber a leitura atual
```

```
    # Lê a comunicação serial
    dados = comunicacaoSerial.readline()

    # trata a string para obter somente os valores desejados
    string_a_tratar = str(dados)
    temp_and_hum = string_a_tratar.split("e")
    umidade = temp_and_hum[0].split("b'")
    temperatura = temp_and_hum[1].split("\\r")

    umidade_float = float(umidade[1])
    temperatura_float = float(temperatura[0])

    # formata para 1 casa decimal
    humidity = '{0:0.1f}'.format(umidade_float)
    temperature = '{0:0.1f}'.format(temperatura_float)
    print(humidity)
    print(temperature)

    templateData = {
       'temperature': temperature,
       'humidity': humidity
    }
    # chama o html mandando os parâmetros do sensor
    return render_template('index.html', **templateData)


if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```