CLÁUDIO LUÍS VIEIRA OLIVEIRA
HUMBERTO AUGUSTO PIOVESANA ZANETTI

PROJETOS COM **PYTHON E ARDUINO**

COMO DESENVOLVER PROJETOS PRÁTICOS DE ELETRÔNICA, AUTOMAÇÃO E IOT

érica

RESOLUÇÃO DOS EXERCÍCIOS

CAPÍTULO 1 – INTRODUÇÃO AO PYTHON

1.

```
tensao = float(input('Digite a tensão: '))
corrente = float(input('Digite a corrente: '))
resistencia = tensao / corrente
print('A resistência é', resistencia, 'Ohms')
```

cap1-ex1.py

2.

```
n1 = input ('Digite o primeiro nome: ')
n2 = input ('Digite o segundo nome: ')
n3 = input ('Digite o terceiro nome: ')
if n1 < n2 and n1 < n3:
  print (n1)
  if n2 < n3:
    print (n2)
    print (n3)
  else:
    print (n3)
    print (n2)
elif n2 < n3:
  print(n2)
  if n1 < n3:
    print (n1)
    print (n3)
  else:
    print (n3)
    print (n1)
else:
  print(n3)
```

```
  if n1 < n2:
    print (n1)
    print (n2)
  else:
    print (n2)
    print (n1)
```

cap1-ex2.py

3.

```
i = 1
while i <= 10:
  num = float(input('Digite um número: '))
  if i > 1:
    if num < menor:
      menor = num
    if num > maior:
      maior = num
  else:
    menor = num
    maior = num
  i = i + 1
dif = maior - menor
print('A diferença entre o maior e o menor é', dif)
```

cap1-ex3.py

4.

```
r1 = r2 = r3 = 1
while r1 != 0 and r2 != 0 and r3 != 0:
  r1 = float(input('Digite o valor do R1: '))
  r2 = float(input('Digite o valor do R2: '))
  r3 = float(input('Digite o valor do R3: '))
  serie = r1 + r2 + r3
  print('O valor em série da associação é', serie, 'Ohms.')
```

cap1-ex4.py

5.

```
valores = []
i = 0
while i < 10:
  valores.append(float(input('Digite um número: ')))
  i = i + 1
i = 9
while i >= 0:
  print(valores[i], end=', ')
  i = i - 1
```

cap1-ex5.py

CAPÍTULO 3 – ENTRADAS E SAÍDAS DIGITAIS

1.

```
from pyfirmata import Arduino

arduino = Arduino('especificar_porta_serial')
led1 = arduino.get_pin('d:12:o')
led2 = arduino.get_pin('d:13:o')

while True:
  led1.write(1)
  led2.write(0)
  arduino.pass_time(0.25)
  led1.write(0)
  led2.write(1)
  arduino.pass_time(0.25)
```

cap3-ex1.py

2.

```
from pyfirmata import Arduino, INPUT, OUTPUT, util

arduino = Arduino('especificar_porta_serial')
led1 = 12
led2 = 13
botao = 2
anterior = False
estado = False

it = util.Iterator(arduino)
it.start()

arduino.digital[botao].mode = INPUT
arduino.digital[led1].mode = OUTPUT
arduino.digital[led2].mode = OUTPUT

while True:
  valor = arduino.digital[botao].read()
  if valor == True and anterior == False:
    estado = not estado
    arduino.digital[led1].write(estado)
    arduino.digital[led2].write(not estado)
  anterior = valor
  arduino.pass_time(0.05)
```

cap3-ex2.py

3.

```
from pyfirmata import Arduino, OUTPUT

arduino = Arduino('especificar_porta_serial')
arduino.digital[13].mode = OUTPUT

vezes = input('Especifique quantas vezes o LED deverá piscar: ')
i = 1

while i <= vezes:
  arduino.digital[13].write(1)
  arduino.pass_time(0.5)
  arduino.digital[13].write(0)
  arduino.pass_time(0.5)
  i = i + 1
```

cap3-ex3.py

4.

```
from pyfirmata import Arduino, OUTPUT

arduino = Arduino('especificar_porta_serial')
arduino.digital[10].mode = OUTPUT
arduino.digital[11].mode = OUTPUT
arduino.digital[12].mode = OUTPUT
arduino.digital[13].mode = OUTPUT

while True:
  qtd = input('Digite quantos LEDs devem acender ou 0 para desligar
todos: ')
  if qtd == '0':
    arduino.digital[10].write(0)
    arduino.digital[11].write(0)
    arduino.digital[12].write(0)
    arduino.digital[13].write(0)
  elif qtd == '1':
    arduino.digital[10].write(1)
    arduino.digital[11].write(0)
    arduino.digital[12].write(0)
    arduino.digital[13].write(0)
  elif qtd == '2':
    arduino.digital[10].write(1)
    arduino.digital[11].write(1)
    arduino.digital[12].write(0)
    arduino.digital[13].write(0)
  elif qtd == '3':
    arduino.digital[10].write(1)
    arduino.digital[11].write(1)
    arduino.digital[12].write(1)
    arduino.digital[13].write(0)
  elif qtd == '4':
    arduino.digital[10].write(1)
    arduino.digital[11].write(1)
    arduino.digital[12].write(1)
    arduino.digital[13].write(1)
  else:
    print('ERRO: Digite apenas 0, 1, 2, 3 ou 4! ')
```

cap3-ex4.py

5.

```
from pyfirmata import Arduino, INPUT, util

arduino = Arduino('especificar_porta_serial')
botao = 2
contador = 0
anterior = False
estado = False

it = util.Iterator(arduino)
it.start()

arduino.digital[botao].mode = INPUT

while True:
  valor = arduino.digital[botao].read()
  if valor == True and anterior == False:
    estado = not estado
    contador = contador + 1
    print('A chave táctil foi pressionada', contador, 'vezes.')
  anterior = valor
  arduino.pass_time(0.05)
```

cap3-ex5.py

CAPÍTULO 4 – ENTRADAS E SAÍDAS ANALÓGICAS

1.

```
from pyfirmata import Arduino, PWM

arduino = Arduino('especificar_porta_serial')
vermelho = 9
azul = 6
verde = 5

arduino.digital[vermelho].mode = PWM
arduino.digital[azul].mode = PWM
arduino.digital[verde].mode = PWM

while True:
  cor = input ('Digite a cor que deverá ser acesa? ')
  if cor = 'vermelho':
    arduino.digital[vermelho].write(1.0)
    arduino.digital[verde].write(0.0)
    arduino.digital[azul].write(0.0)
  elif cor = 'verde':
    arduino.digital[vermelho].write(0.0)
    arduino.digital[verde].write(1.0)
    arduino.digital[azul].write(0.0)
  elif cor = 'azul':
    arduino.digital[vermelho].write(0.0)
    arduino.digital[verde].write(0.0)
    arduino.digital[azul].write(1.0)
```

```
  elif cor = 'apagar':
    arduino.digital[vermelho].write(0.0)
    arduino.digital[verde].write(0.0)
    arduino.digital[azul].write(0.0)
  else:
    print('Digite apenas vermelho, verde, azul ou apagar!')
  arduino.pass_time(0.2)
```

cap4-ex1.py

2.

```
from pyfirmata import Arduino, util
from math import log

def obter_temp_celsius (valor):
  tempK = log(10000.0 * (1.0 / valor - 1))
  tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK *
tempK )) * tempK)
  tempC = tempK - 273.15
  return tempC

arduino = Arduino('especificar_porta_serial')
it = util.Iterator(arduino)
it.start()
termistor = arduino.get_pin('a:0:i')
termistor.enable_reporting()
vermelho = arduino.get_pin('d:12:o')
verde = arduino.get_pin('d:13:o')

while True:
  valor = str(termistor.read())
  if valor != 'None':
    valor = float(valor)
    print (round(obter_temp_celsius(valor), 1), '°C')
    if valor < 30:
      verde.write(1)
      vermelho.write(0)
    else:
      verde.write(0)
      vermelho.write(1)
    arduino.pass_time(0.2)
```

cap4-ex2.py

3.

```
from pyfirmata import Arduino, util

arduino = Arduino('especificar_porta_serial')
it = util.Iterator(arduino)
it.start()

ldr = arduino.get_pin('a:0:i')
led1 = arduino.get_pin('d:12:o')
led2 = arduino.get_pin('d:13:o')

ldr.enable_reporting()
while True:
  valor = str(ldr.read())
```

```
  print (valor)
  if valor != 'None':
    valor = float(valor)
    if valor < 0.5:
      led1.write(1)
      led2.write(1)
    else:
      led1.write(0)
      led2.write(0)
    arduino.pass_time(0.1)
```

cap4-ex3.py

4.

```
from pyfirmata import Arduino, util

arduino = Arduino('especificar_porta_serial')
it = util.Iterator(arduino)
it.start()

pot = arduino.get_pin('a:0:i')
led1 = arduino.get_pin('d:11:o')
led2 = arduino.get_pin('d:12:o')
led3 = arduino.get_pin('d:13:o')

pot.enable_reporting()
while True:
  valor = str(pot.read())
  print (valor)
  if valor != 'None':
    valor = float(valor)
    if valor < 0.25:
      led1.write(0)
      led2.write(0)
      led3.write(0)
    elif valor < 0.5:
      led1.write(1)
      led2.write(0)
      led3.write(0)
    elif valor < 0.75:
      led1.write(1)
      led2.write(1)
      led3.write(0)
    else:
      led1.write(1)
      led2.write(1)
      led3.write(1)
    arduino.pass_time(0.1)
```

cap4-ex4.py

5.

```
from pyfirmata import Arduino, PWM, util

import random

arduino = Arduino('especificar_porta_serial')
potenciometro = 0
vermelho = 9
azul = 6
verde = 5

arduino.digital[vermelho].mode = PWM
arduino.digital[azul].mode = PWM
arduino.digital[verde].mode = PWM

it = util.Iterator(arduino)
it.start()
arduino.analog[potenciometro].enable_reporting()

while True:
  arduino.digital[vermelho].write(random.random())
  arduino.digital[azul].write(random.random())
  arduino.digital[verde].write(random.random())
  valor = str(arduino.analog[potenciometro].read())
  if valor != 'None':
    arduino.pass_time(float(valor) * 2.0)
```

cap4-ex5.py

CAPÍTULO 5 – UM POUCO MAIS SOBRE LISTAS E DICIONÁRIOS

1.

```
from pyfirmata import Arduino, INPUT, OUTPUT, util
from datetime import datetime

lista = []
arduino = Arduino('especificar_porta_serial')
botao = 2
anterior = False
estado = False

it = util.Iterator(arduino)
it.start()

arduino.digital[botao].mode = INPUT

while True:
  valor = arduino.digital[botao].read()
  if valor == True and anterior == False:
    estado = not estado
    agora = datetime.now()
    lista.append(agora.strftime("%d/%m/%Y %H:%M:%S"))
    if len(lista) > 5:
      lista.remove(lista[0])
  anterior = valor
  arduino.pass_time(0.05)
```

cap5-ex1.py

2.

```
from pyfirmata import Arduino, util
from math import log
from datetime import datetime

def obter_temp_celsius (valor):
  tempK = log(10000.0 * (1.0 / valor - 1))
  tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK *
tempK )) * tempK)
  tempC = tempK - 273.15
  return tempC

temperaturas = []
arduino = Arduino('especificar_porta_serial')
it = util.Iterator(arduino)
it.start()
termistor = arduino.get_pin('a:0:i')
termistor.enable_reporting()
while True:
  valor = str(termistor.read())
  if valor != 'None':
    valor = float(valor)
    print (round(obter_temp_celsius(valor), 1), '°C')
    temperaturas.append(round(obter_temp_celsius(valor), 1))
    if len(temperaturas) > 10:
      temperaturas.remove(temperaturas[0])
    arduino.pass_time(5.0)
```

cap5-ex2.py

3.

```
from pyfirmata import Arduino, util
from math import log
from datetime import datetime

leituras = []
arduino = Arduino('especificar_porta_serial')
it = util.Iterator(arduino)
it.start()
arduino.analog[0].enable_reporting()
while True:
  valor = str(arduino.analog[0].read())
  print (valor)
  if valor != 'None':
    valor = float(valor)
    agora = datetime.now()
    leituras.append({'data_hora': agora.strftime("%d/%m/%Y %H:%M:%S"),
'valor': valor})
    if len(leituras) > 10:
      leituras.remove(leituras[0])
    arduino.pass_time(0.1)
```

cap5-ex3.py

4.

```
from pyfirmata import Arduino, OUTPUT

arduino = Arduino('especificar_porta_serial')
led1 = 10
led2 = 11
led3 = 12
led4 = 13
num  =  0

digito = [
  [ 0, 0, 0, 0 ],
  [ 0, 0, 0, 1 ],
  [ 0, 0, 1, 0 ],
  [ 0, 0, 1, 1 ],
  [ 0, 1, 0, 0 ],
  [ 0, 1, 0, 1 ],
  [ 0, 1, 1, 0 ],
  [ 0, 1, 1, 1 ],
  [ 1, 0, 0, 0 ],
  [ 1, 0, 0, 1 ],
  [ 1, 0, 1, 0 ],
  [ 1, 0, 1, 1 ],
  [ 1, 1, 0, 0 ],
  [ 1, 1, 0, 1 ],
  [ 1, 1, 1, 0 ],
  [ 1, 1, 1, 1 ]
]

arduino.digital[led1].mode = OUTPUT
arduino.digital[led2].mode = OUTPUT
arduino.digital[led3].mode = OUTPUT
arduino.digital[led4].mode = OUTPUT

while True:
  arduino.digital[led1].write(digito[num][0])
  arduino.digital[led2].write(digito[num][1])
  arduino.digital[led3].write(digito[num][2])
  arduino.digital[led4].write(digito[num][3])
  arduino.pass_time(1.0)
  num = num + 1
  if num > 15:
    num = 0
```

cap5-ex4.py

5.

```
from pyfirmata import Arduino, PWM
import random

arduino = Arduino('especificar_porta_serial')
valores = []
vermelho = 9
azul = 6
verde = 5

arduino.digital[vermelho].mode = PWM
arduino.digital[azul].mode = PWM
```

```
arduino.digital[verde].mode = PWM

while True:
  r = random.random()
  g = random.random()
  b = random.random()
  arduino.digital[vermelho].write(r)
  arduino.digital[azul].write(b)
  arduino.digital[verde].write(g)
  valores.append({'vermelho': r, 'verde': g, 'azul': b})
  if len(valores) > 10:
    valores.remove(valores[0])
  arduino.pass_time(2.0)
```

cap5-ex5.py

CAPÍTULO 6 – DISPLAYS

1.

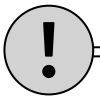
```
from pyfirmata import Arduino

anterior = False
pos = 0

# Irá exibir Pyt (as três primeiras letras de Python)
caracter = [
  [ 1, 1, 0, 0, 1, 1, 1 ],
  [ 0, 1, 1, 1, 0, 1, 1 ],
  [ 0, 0, 0, 1, 1, 1, 1 ]
]

arduino = Arduino('especificar_porta_serial')

seg = list()
seg.append(arduino.get_pin('d:7:o'))
seg.append(arduino.get_pin('d:8:o'))
seg.append(arduino.get_pin('d:9:o'))
seg.append(arduino.get_pin('d:10:o'))
seg.append(arduino.get_pin('d:11:o'))
seg.append(arduino.get_pin('d:12:o'))
seg.append(arduino.get_pin('d:13:o'))

def exibir(posicao):
  for i in range(0, 7):
    seg[i].write(caracter[posicao][i])

while True:
  exibir(pos)
  pos = pos + 1
  if pos > 2:
    pos = 0
  arduino.pass_time(1.0)
```

cap6-ex1.py

2.

```
from pyfirmata import Arduino

digito = [
  [ 1, 1, 1, 1, 1, 1, 0 ],
  [ 0, 1, 1, 0, 0, 0, 0 ],
  [ 1, 1, 0, 1, 1, 0, 1 ],
  [ 1, 1, 1, 1, 0, 0, 1 ],
  [ 0, 1, 1, 0, 0, 1, 1 ],
  [ 1, 0, 1, 1, 0, 1, 1 ],
  [ 1, 0, 1, 1, 1, 1, 1 ],
  [ 1, 1, 1, 0, 0, 0, 0 ],
  [ 1, 1, 1, 1, 1, 1, 1 ],
  [ 1, 1, 1, 0, 0, 1, 1 ]
]

arduino = Arduino('especificar_porta_serial')
num = 0

seg = list()
seg.append(arduino.get_pin('d:7:o'))
seg.append(arduino.get_pin('d:8:o'))
seg.append(arduino.get_pin('d:9:o'))
seg.append(arduino.get_pin('d:10:o'))
seg.append(arduino.get_pin('d:11:o'))
seg.append(arduino.get_pin('d:12:o'))
seg.append(arduino.get_pin('d:13:o'))

def exibir(numero):
  for i in range(0, 7):
    seg[i].write(digito[numero][i])

while True:
  num = int(input('Digite um número entre 0 e 9:'))
  if valor >= 0 and valor < 10:
    exibir(num)
  else:
    print('Digite apenas números entre 0 e 9!')
```

cap6-ex2.py



ATENÇÃO!

Para resolver os exercícios 3 a 5, deve-se realizar as alterações no StandardFirmata e adicionar as funções em Python para uso do LCD, conforme descrito no projeto “Display de cristal líquido”.

3.

```
from pyfirmata import Arduino
import lcd

arduino = Arduino('especificar_porta_serial')

lcd.escrever(arduino, 0, 0, 'Nome')
lcd.escrever(arduino, 0, 1, 'Sobrenome')
arduino.pass_time(5.0)
lcd.limpar(arduino)
arduino.exit()
```

cap6-ex3.py

4.

```
from pyfirmata import Arduino
import lcd

arduino = Arduino('especificar_porta_serial')

nome = input('Digite o nome: ')
sobrenome = input('Digite o sobrenome: ')
lcd.escrever(arduino, 0, 0, nome)
lcd.escrever(arduino, 0, 1, sobrenome)
arduino.pass_time(5.0)
lcd.limpar(arduino)
arduino.exit()
```

cap6-ex4.py

5.

```
from pyfirmata import Arduino, util, INPUT
from datetime import datetime
import lcd
from math import log

def obter_temp_kelvin (valor):
  tempK = log(10000.0 * (1.0 / valor - 1))
  tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK *
tempK)) * tempK
  return tempK

def obter_temp_celsius (valor):
  tempC = obter_temp_kelvin (valor) - 273.15
  return tempC

def obter_temp_fahrenheit (valor):
  tempF = (obter_temp_kelvin (valor) - 273.15) * 9 / 5 + 32
  return tempF

mes_extenso = ['Janeiro', 'Fevereiro', 'Marco',
  'Abril', 'Maio', 'Junho', 'Julho', 'Agosto',
  'Setembro', 'Outubro', 'Novembro', 'Dezembro']
dia_semana = ['Segunda', 'Terca', 'Quarta',
  'Quinta', 'Sexta', 'Sabado', 'Domingo']
anterior = False
```

```
funcao  =  0
linha = ['', '']

arduino = Arduino('especificar_porta_serial')
botao = arduino.get_pin('d:6:i')
termistor = arduino.get_pin('a:0:i')
termistor.enable_reporting()
it = util.Iterator(arduino)
it.start()

temp = 0

while True:
  agora = datetime.now()
  valor = botao.read()

  if valor == True and anterior == False:
    lcd.limpar(arduino)
    funcao = funcao + 1
    if funcao > 1:
      funcao = 0

  if funcao == 0:
    linha[0] = str(agora.day) + '/' + str(agora.month) + '/' +
str(agora.year)
    linha[1] = str(agora.hour) + ':' + str(agora.minute).zfill(2) + ':' +
str(agora.second).zfill(2)
  else:
    linha[0] = mes_extenso[agora.month - 1] + ' ' + str(agora.day)
    linha[1] = dia_semana[agora.weekday()]

  for i in range(2):
    lcd.escrever(arduino, 0, i, linha[i])

  temperatura = str(termistor.read())
  if temperatura != 'None':
    temperatura = float(temperatura)
    if temp == 0:
      temperatura = round(obter_temp_celsius(temperatura), 1)
      lcd.escrever(arduino, 10, 1, str(temperatura) + "'C")
    elif temp == 1:
      temperatura = round(obter_temp_fahrenheit(temperatura), 1)
      lcd.escrever(arduino, 10, 1, str(temperatura) + "'F")
    else:
      temperatura = round(obter_temp_kelvin(temperatura), 1)
      lcd.escrever(arduino, 10, 1, str(temperatura) + "K")
     temp = temp + 1
     if temp > 2:
       temp = 0
  anterior = valor
  arduino.pass_time(1.0)
```

cap6-ex5.py

CAPÍTULO 7 – INTERFACE GRÁFICA COM PYTHON

1.

```
from tkinter import *

def muda_texto():
     label['text'] = 'Mudou!'
     label['fg'] = 'red'

janela = Tk()
janela.title("Aplicação em Tkinter")
janela.geometry("300x50")

frame = Frame(master=janela)
frame.pack()

label = Label(master=frame, text="Clique no botão e mude o texto")
label.pack()

botao = Button(master=frame, text="Mudar",command=muda_texto)
botao.pack()

janela.mainloop()
```

cap7-ex1.py

2.

```
from tkinter import *

def mostrar_texto():
  valor = entrada.get()
  labelsaida['text'] = valor

def apagar_texto():
  labelsaida['text'] = ' '

janela = Tk()
janela.title("Aplicação em Tkinter")
janela.geometry("300x70")

frame = Frame(master=janela)
frame.pack()

labelentrada = Label(master=frame, text="Digite um número:")
labelentrada.grid(row=0, column=0)

entrada=Entry(master=frame,width=10)
entrada.grid(row=0, column=1)

botao1 = Button(master=frame, text="Mostrar",command=mostrar_texto)
botao.grid(row=1, column=0)

botao2 = Button(master=frame, text="Apagar",command=apagar_texto)
botao.grid(row=1, column=1)
```

```
labelsaida = Label(master=frame)
labelsaida.grid(row=2, column=0)

janela.mainloop()
```

cap7-ex2.py

3.

```
from pyfirmata import Arduino, OUTPUT
from tkinter import *

arduino = Arduino('especificar_porta_serial')
arduino.digital[13].mode = OUTPUT
arduino.digital[12].mode = OUTPUT


def controle(pino,valor):
    arduino.digital[pino].write(valor)

janela = Tk()
janela.title("Acender e apagar LED com botão")
janela.geometry("350x60")

frame = Frame(master=janela)
frame.pack()

botaoacendeled1 = Button(master=frame, text="Acender LED
1",command=lambda:controle(13,1))
botaoacende.grid(row=0, column=0)

botaoapagaled1 = Button(master=frame, text="Apagar LED
1",command=lambda:controle(13,0))
botaoapaga.grid(row=0, column=1)

botaoacendeled2 = Button(master=frame, text="Acender LED
2",command=lambda:controle(12,1))
botaoacende.grid(row=1, column=0)

botaoapagaled2 = Button(master=frame, text="Apagar LED
2",command=lambda:controle(12,0))
botaoapaga.grid(row=1, column=1)


janela.mainloop()
```

cap7-ex3.py

4.

```
from pyfirmata import Arduino, PWM
from tkinter import *

arduino = Arduino('especificar_porta_serial')
arduino.digital[10].mode = PWM
arduino.digital[9].mode = PWM
```

```
def controle_led1 (intensidade):
    arduino.digital[10].write(int(intensidade)/100)
    print(intensidade)

def controle_led2 (intensidade):
    arduino.digital[9].write(int(intensidade)/100)
    print(intensidade)


janela = Tk()
janela.title("Controle de luminosidade do LED")
janela.geometry("400x50")

frame = Frame(master=janela)
frame.pack()

slider1 = Scale(master = frame, length = 300, from_= 0, to = 100, orient
= HORIZONTAL, command = controle_led1)
slider.grid(row=0, column=0)

slider2 = Scale(master = frame, length = 300, from_= 0, to = 100, orient
= HORIZONTAL, command = controle_led2)
slider.grid(row=1, column=0)


janela.mainloop()
```

cap7-ex4.py

5.

```
from pyfirmata import Arduino, util
from math import log
from tkinter import *
from tkinter import ttk

arduino = Arduino('COM9')
it = util.Iterator(arduino)
it.start()

arduino.analog[0].enable_reporting()

janela = Tk()
janela.title("Mostrando a temperatura")
janela.geometry("500x300")

frame = Frame(master=janela)
frame.pack()

luminosidade = StringVar()

def mostra_luminosidade():
  valor = str(arduino.analog[0].read())
  if valor != 'None':
    valor = float(valor)
    progressbar["value"]=valor
    janela.after(500, mostra_luminosidade)

progressbar = ttk.Progressbar(frame, style='color.Horizontal.
TProgressbar', orient="vertical",length=220,mode="determinate")
```

```
progressbar.grid(row=0, column=0)

label=Label(frame, textvariable=temperatura, font=("Helvetica", 20))
label.grid(row=0, column=1)

janela.protocol('WM_DELETE_WINDOW', janela.destroy)
janela.after(500,mostra_luminosidade)
janela.mainloop()
```

cap7-ex5.py

CAPÍTULO 8 – INTERNET DAS COISAS

1. Inicialmente, deve-se definir a classe singleton, conforme descrito no projeto "Controle do LED através da internet".

```
from arduino import Arduino
from flask import Flask, render_template
import datetime

LED1 = 12
LED2 = 13
estado = {'led1' : False, 'led2' : False}

app = Flask(__name__)

@app.route('/')
def inicio():
  return mostra_estado()

@app.route('/led1/1')
def ligar_led1():
  arduino = Arduino()
  arduino.digitalWrite(LED1, 1)
  estado['led1'] = True
  return mostra_estado()

@app.route('/led1/0')
def desl_led1():
  arduino = Arduino()
  arduino.digitalWrite(LED1, 0)
  estado['led1'] = False
  return mostra_estado()

@app.route('/led2/1')
def ligar_led2():
  arduino = Arduino()
  arduino.digitalWrite(LED2, 1)
  estado['led2'] = True
  return mostra_estado()

@app.route('/led2/0')
def desl_led2():
  arduino = Arduino()
  arduino.digitalWrite(LED2, 0)
  estado['led2'] = False
  return mostra_estado()
```

```
def mostra_estado():
  return render_template('cap8-ex1.html', **estado)

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

cap8-ex1.py

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Controle de LEDs</title>
  </head>
  <body>
    <h2>Controle de LEDs</h2>
    O LED1 está
    {% if led1 %}
      ligado.
      <button type="button"
      onclick="window.location.href='/led1/0';">
      Desligar</button>
    {% else %}
      desligado.
      <button type="button"
      onclick="window.location.href='/led1/1';">
      Ligar</button>
    {% endif %}
    <br>
    O LED2 está
    {% if led2 %}
      ligado.
      <button type="button"
      onclick="window.location.href='/led2/0';">
      Desligar</button>
    {% else %}
      desligado.
      <button type="button"
      onclick="window.location.href='/led2/1';">
      Ligar</button>
    {% endif %}
  </body>
</html>
```

cap8-ex1.html

2. Inicialmente, deve-se definir a classe singleton, conforme descrito no projeto "Temperatura web".

```
from flask import Flask, render_template
from arduino import Arduino
from math import log
import datetime

def obter_temp_kelvin (valor):
  tempK = log(10000.0 * (1.0 / valor - 1))
  tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK *
tempK)) * tempK)
```

```
  return tempK

def obter_temp_celsius (valor):
  tempC = obter_temp_kelvin (valor) - 273.15
  return tempC

def obter_temp_fahrenheit (valor):
  tempF = (obter_temp_kelvin (valor) - 273.15) * 9 / 5 + 32
  return tempF

dado = {'tempC' : 0, 'tempF': 0, 'tempK': 0}

app = Flask(__name__)

@app.route('/')
def inicio():
  arduino = Arduino()
  valor = arduino.analogRead(0)
  if valor >= 0.0 and valor <= 1.0:
    dado['tempC'] = round(obter_temp_celsius(valor), 1)
    dado['tempF'] = round(obter_temp_fahrenheit(valor), 1)
    dado['tempK'] = round(obter_temp_kelvin(valor), 1)
  else:
    dado['tempC'] = '...'
    dado['tempF'] = '...'
    dado['tempK'] = '...'
  return render_template('cap8-ex2.html', **dado)

if __name__ == '__main__':
  app.run(debug=True)
```

cap8-ex2.py

```
<!DOCTYPE html>
<html>
  <head>
    <title>Temperatura</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/
jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/
umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/
bootstrap.min.js"></script>
  </head>
  <body>
    <div class="container">
      <div class="jumbotron">
        <h1>Temperatura</h1>
      </div>
      <h1>
```

```
        <img src="static/images/termometro.png" align="left">
        <span id="tempC">{{ tempC }} </span>°C<br>
        <span id="tempF">{{ tempF }} </span>°F<br>
        <span id="tempK">{{ tempK }} </span>K
      </h1>
    </div>
  </body>
</html>
```

cap8-ex2.html

3. Inicialmente, deve-se definir a classe singleton, conforme descrito no projeto "Temperatura web".

```
from flask import Flask, render_template
from arduino import Arduino

dado = {'imagem' : ''}

app = Flask(__name__)

@app.route('/')
def inicio():
  arduino = Arduino()
  valor = arduino.analogRead(0)
  if valor < 0.5:
    dado['imagem'] = 'lua.png'
  else:
    dado['imagem'] = 'sol.png'
  return render_template('cap8-ex3.html', **dado)

if __name__ == '__main__':
  app.run(debug=True)
```

cap8-ex3.py

```
<!DOCTYPE html>
<html>
  <head>
    <title>Dia ou Noite</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/
jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/
umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/
bootstrap.min.js"></script>
  </head>
  <body>
    <div class="container">
```

```
      <div class="jumbotron">
        <h1>Dia ou Noite!</h1>
      </div>
      <h1>
        <img src="static/images/{{ imagem }}">
      </h1>
    </div>
  </body>
</html>
```

cap8-ex3.html

4. Inicialmente, deve-se definir a classe singleton, conforme descrito no projeto "Controle do LED através da internet".

```
from arduino import Arduino
from flask import Flask, render_template
import datetime

R = 9
G = 5
B = 6
estado = {'R': False, 'G': False, 'B': False}

app = Flask(__name__)

@app.route('/')
def inicio():
  return mostra_estado()

@app.route('/R/1')
def ligar_vermelho():
  arduino = Arduino()
  arduino.digitalWrite(R, 1)
  estado['R'] = True
  return mostra_estado()

@app.route('/R/0')
def desl_vermelho():
  arduino = Arduino()
  arduino.digitalWrite(R, 0)
  estado['R'] = False
  return mostra_estado()

@app.route('/G/1')
def ligar_verde():
  arduino = Arduino()
  arduino.digitalWrite(G, 1)
  estado['G'] = True
  return mostra_estado()

@app.route('/G/0')
def desl_verde():
  arduino = Arduino()
  arduino.digitalWrite(G, 0)
  estado['G'] = False
  return mostra_estado()
```

```
@app.route('/B/1')
def ligar_azul():
  arduino = Arduino()
  arduino.digitalWrite(B, 1)
  estado['B'] = True
  return mostra_estado()

@app.route('/B/0')
def desl_azul():
  arduino = Arduino()
  arduino.digitalWrite(B, 0)
  estado['B'] = False
  return mostra_estado()

def mostra_estado():
  return render_template('cap8-ex4.html', **estado)

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

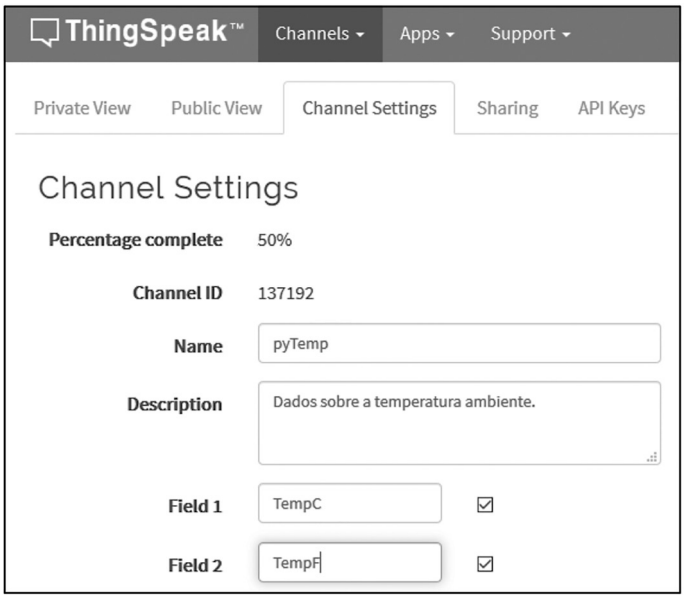
cap8-ex4.py

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Controle do LED RGB</title>
  </head>
  <body>
    <h2>Controle do LED RGB</h2>
    O LED Vermelho está
    {% if R %}
      ligado.
      <button type="button"
      onclick="window.location.href='/R/0';">
      Desligar</button>
    {% else %}
      desligado.
      <button type="button"
      onclick="window.location.href='/R/1';">
      Ligar</button>
    {% endif %}
    <br>
    O LED Verde está
    {% if G %}
      ligado.
      <button type="button"
      onclick="window.location.href='/G/0';">
      Desligar</button>
    {% else %}
      desligado.
      <button type="button"
      onclick="window.location.href='/G/1';">
      Ligar</button>
    {% endif %}
    <br>
    O LED Azul está
    {% if B %}
      ligado.
      <button type="button"
      onclick="window.location.href='/B/0';">
```

```
      Desligar</button>
    {% else %}
      desligado.
      <button type="button"
      onclick="window.location.href='/B/1';">
      Ligar</button>
    {% endif %}
  </body>
</html>
```

cap8-ex4.html

5. Acesse a ThingSpeak e edite o canal PyTemp, que foi criado no projeto "ThingSpeak – plataforma para aplicações IoT", por meio da aba *Channel Settings*, conforme mostra a imagem a seguir:



Após realizar as alterações, clique no botão Save Channel.

```
import http.client
from pyfirmata import Arduino, util
from math import log

def obter_temp_kelvin (valor):
  tempK = log(10000.0 * (1.0 / valor - 1))
  tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK *
tempK )) * tempK)
  return tempK

def obter_temp_celsius (valor):
  tempC = obter_temp_kelvin (valor) - 273.15
  return tempC
```

```
def obter_temp_fahrenheit (valor):
  tempF = (obter_temp_kelvin (valor) - 273.15) * 9 / 5 + 32
  return tempF

CHAVE = 'especificar_chave'
SERVIDOR = 'api.thingspeak.com'
PORTA = 'especificar_porta_serial'

arduino = Arduino(PORTA)
it = util.Iterator(arduino)
it.start()
termistor = arduino.get_pin('a:0:i')
termistor.enable_reporting()

while True:
  valor = str(termistor.read())
  if valor != 'None':
    tempC = round(obter_temp_celsius(float(valor)), 1)
    tempF = round(obter_temp_fahrenheit(float(valor)), 1)
    url = '/update?api_key=' + CHAVE + '&field1=' + str(tempC) +
'&field2=' + str(tempF)
    print(url)
    con = http.client.HTTPSConnection(SERVIDOR)
    con.request('GET', url)
    resp = con.getresponse()
    print(resp.status, resp.reason)
    arduino.pass_time(60.0)
```

cap8-ex5.py