RESUMO P1 MICROCONTROLADORES

Usando modo simulador no Thonny

from extra.playground import rodar

@rodar

def main():

Introdução

→ Módulo Turtle:

from turtle import *; from json import load; mainloop() no final do código;

forward(100); right(100); left(100); goto(50, 50); penup(); pendown(); onscreenclick(func); circle(r); write("Hello" + str(x)); fillcolor("blue"); begin_fill(); end_fill();

abrir json → dic = load(open('paises.json', encoding="UTF-8"));

ler da tela → pais = textinput("Digite o nome de um país", "País");

define ângulo como 0º, independente do ângulo inicial→ setheading(0)

espiral → for i in range(100): forward(i/5); right(10);

Projeto 01

from time import sleep; from os import system;

sempre checar se metadada (e suas chaves) são != None

while True sempre última coisa do código!

pausa programa até que botão seja pressionado → botão.wait_for_press()

tocar 2x mais lento → player.speed = -2

zeros à esquerda → lcd.message("\n{:02d}".format(minutos))

embaralhar e tocar lista → with open('playlist.txt') as f: lines = f.readlines(); random.shuffle(lines); with open("shuffle.txt", "w") as output: for el in lines: output.write(el);

rolar faixa no lcd →

```
if len(metadados["Title"]) > 16:
    if cont > len(metadados["Title"]):
        cont = 0
    lcd.message(metadados["Title"][cont:cont+16])
    cont += 1
else:
    lcd.message(metadados["Title"])
```

Projeto 02

rotas do Flask sempre devem retornar um texto!

from flask import Flask, redirect;

pagina "blabla.html" deve estar sempre dentro da pasta "templates" para rodar o render_template!

HTML → Negrito; <h3> Titulo </h3>

quebra de linha →

rodar local → localhost:5000/nome_da_rota/param1/param2

rodar ngrok → https://numeroaleatorio.ngrok.io/nome_da_rota/param1/param2

debugar local → app.run(port=5000, debug=True)

html do tamanho da tela de celular (colocar no início do html)→ <meta name="viewport" content="width=device-width, initial-scale=1.5">

passando parâmetros no render_template → return render_template("menu.html", len=len(dicionarios_de_canais), text_lists=dicionarios_de_canais)

escrevendo loop e código python dentro do html →

```
<ul>
   {%for i in range(0, len)%}
      <li><a href="/muda_canal/{{text_lists[i]["código"]}}">{{text_lists[i]["nome"]}}</a></li>
   {%endfor%}
</ul>
```

Projeto 03

sensor de distância sempre em metros!

sempre checar se documento buscado no Mongo é != None ou se documentos é != [ ]

agora – timedelta(days=300, weeks=40, minutes=50); agora.strftime("data %d/%m/%Y às %H:%M:%S"); agora.strftime("%c");

sensor.max_distance = 2 #valor máximo em metros;

busca = { }; ordenacao = [["gênero", ASCENDING], ["artista", DESCENDING]];

acesso local ao Mongo → localhost:1234

outro jeito de inserir → colecao_de_filmes.insert_one(filme1); colecao_de_filmes.insert_many([filme1, filme2]);

comparações de busca →busca = {"chave": valor}; {"chave": {"$ne": valor}}; {"chave": {"$gt": valor}}; {"chave": {"$gte": valor}}; {"chave": {"$lt": valor}}; {"chave": {"$lte": valor}}; {"chave": {"$in": lista}}; {"chave": {"$nin": lista}}

Projeto 04

gravação simultânea à escrita de código →

```
from subprocess import Popen; global aplicativo; aplicativo = None;

def iniciar_gravacao():
        global aplicativo
        aplicativo = Popen("arecord, "--duration", "30", "audio.wav"])

def parar_gravacao():
        global aplicativo
        if aplicativo != None:
                aplicativo.terminate()
                aplicativo = None

iniciar_gravacao()

# faz coisas no código …

parar_gravacao()
```

comandos Windows → ver pp. 22 Teoria 04

enviar audio pro Telegram →

```
endereco = endereco_base + "/sendVoice"; dados = {"chat_id": id_da_conversa}; arquivo = {"voice": open("audio.ogg", "rb")}; resposta = post(endereco, data=dados, files=arquivo); print(resposta.text);
```

enviar áudio sempre no formato ogg!

Busca contínua de novas msgs → ver pp. 53 Teoria 04

Download de arquivo do chat → ver pp. 56 Teoria 04

Dados pessoais Telegram → chave = "5177772037:AAGTaTpNuSj17Ef40unWlMShxcb3jaKvsOQ"

```
endereco_base = "https://api.telegram.org/bot" + chave

id_da_conversa = "5271993060"
```

rolar msg sem acento no display →

```
while true:
        conv = unidecode(texto)
        if len(conv) <= 16:
                lcd.message(conv)
        else:
                for i in range(len(conv)-15):
                        lcd.clear()
                        lcd.message(conv[i:i+16])
                        sleep(0.2)
```

respostas pré-definidas no Telegram →dados = {"chat_id": id_da_conversa, "text": "Olá!", "reply_markup": {"keyboard": [["Abrir"], ["Soar Alarme"], ["Ignorar"]], "one_time_keyboard": True}}

Projeto 05

Timer só pode ser startado uma vez! (deve criar novamente)

Timer recorrente → ver pp. 9 Teoria 05

Alteração gradual de luz → sensor = LightSensor(8, queue_len=20) → claro = 1, escuro = 0;

"value1" no dic do Python → {{Value1}} na msg de resposta do IFTTT;

Ver url do ngrok via terminal → ~ $ ngrok http 5000

Dados pessoais IFTTT → chave = "cOsV4PelmB1EWsDoCKeATe"

```
endereco = "https://maker.ifttt.com/trigger/" + "evento_teste" + "/with/key/"  + chave
```

This passando número e string para rota do flask → ver applet "Google assistant -> web request"

Passar mais parâmetros para o IFTTT → ver applet "Rasp -> Google Sheets":

```
agora = datetime.now()
data = agora - timedelta(minutes=1)
val2 = ""
for i in range(5):
    sec = tot_segundos(i, data)
    val2 += (str(int(sec)) + " ||| ")
val2 = val2[:-5]
dados = {"value1": str(agora), "value2": val2}
resultado = post(endereco, json=dados)
```