

UNIVERSIDADE DE FRANCA



Projeto de Pensamento Computacional:

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

2023

Orientador: Renato Rocha

Integrantes:

Mateus Lima Cançado (RGM: 34053808)

Paulo Eduardo Campos Dos Reis Romualdo (RGM: 32796609)

Pedro Otávio Santos de Almeida (RGM: 33791171)

Gabriel Goulart Fanan (RGM: 32515103)

Eduardo Campos Dos Reis Romualdo (RGM: 32796609)

Gustavo Pereira dos Santos (RGM: 34527931)

José Carlos de Moraes sousa (RGM: 34331182)

Felipe Marafiga Ribeiro (RGM: 33395136)

João Victor Antunes Monteiro Braga (RGM: 34615784)

Beatriz Silvestre Alves (RGM: 34101161)

Resumo

Palavras-chave:

INTRODUÇÃO

O diabetes melito inclui um grupo de doenças metabólicas caracterizadas por hiperglicemia, resultante de defeitos na secreção de insulina e/ou em sua ação.(Gross, Silveiro, Camargo et al.)

O teste oral de tolerância à glicose é o método de referência, considerando-se a presença de diabetes ou tolerância à glicose diminuída quando a glicose plasmática de 2h após a ingestão de 75g de glicose para $> \text{ ou } = 200\text{mg/dl}$ ou $> \text{ ou } = 140$ e $<200\text{mg/dl}$, respectivamente. Quando este teste não puder ser realizado, utiliza-se a medida da glicose plasmática em jejum, considerando-se como diabetes ou glicose alterada em jejum quando os valores forem $> \text{ ou } = 126\text{mg/dl}$ ou $> \text{ ou } = 110$ e $<126\text{mg/dl}$, respectivamente (Gross, Silverio, Camargo et al.).

A Hipertensão arterial sistêmica (HAS) é considerada, ao mesmo tempo, uma doença que é um fator de risco, representando um grande desafio para a saúde pública, pois as doenças cardiovasculares constituem a primeira causa de morte no Brasil. É definida quando encontrados valores pressóricos para pressão arterial sistólica acima de 140 mmHg e diastólica acima 90 mmHg. A pressão arterial limítrofe é aquela com valores sistólicos entre 130-139 mmHg e diastólicos entre 85-89 mmHg, enquanto que a pressão arterial normal sistólica < 130 mmHg e diastólica < 85 mm Hg. Já para a pressão arterial classificada como ótima, a pressão arterial sistólica deve estar <120 mmHg e diastólica <80 mmHg (MAGRINI , MARTINI).

JUSTIFICATIVA

OBJETIVOS

O presente projeto tem como objetivo geral o estudo de um robô capaz de fazer medição da pressão do sangue, além de avisar o usuário quando o nível de diabetes ou da pressão do sangue está acima do normal, podendo ter um tipo de assistente como Google ou Alexa .

MATERIAIS E MÉTODOS

O presente trabalho foi realizado através de literatura específica, robótica, algoritmo e programação em Python.

Materiais necessários para o protótipo

- Um glicosímetro
- Um Esfigmomanômetro
- Uma assistente virtual portátil e programável (Alexa, Google Assistente, Cortana, Bing, Chat GPT (recomendado) etc...)

Um sistema térmico

Alarmes

Um sistema de segurança

Um sistema operacional (windows, Mac, Linux etc...)

Software próprio

Tela TouchScreen (no mínimo 9 polegadas)

Temporizador

Alto Falantes

Área de acessibilidade para portadores de deficiências diversas (cegos, mudos, surdos, etc...)

SAC (Serviço de Atendimento ao Cliente)

Plataformas. Essas peças são fundamentais para iniciar a montagem do projeto. ...

Vigas...

Conectores...

Engrenagem...

Circuito Buzzer...

Parafusos e porcas....

Arduino uno R3...

Placa de ensaio pequena...

1 Resistor 220 ons...

2 Acionador de motor de ponte h

Potenciômetro...

3 Sensores de distância ultra sônicos...

2 Regulador de tensão XL 4005

E 1 Sensor de temperatura (TMP36)...

(o número de cada material pode variar com o tamanho do protótipo, por isso quanto maior o protótipo maior vai ser o número de materiais)

TEMPO DE PRODUÇÃO

Organograma de produção CRIAÇÃO DO PRIMEIRO PROTÓTIPO



ESTIMATIVA DE TEMPO



Como o nosso robô é capaz de fazer medições e enviar alertas em tempo real, ele pode ser uma ferramenta muito útil em um ambiente hospitalar, onde a monitorização constante da pressão arterial e dos níveis de açúcar no sangue são cruciais para o tratamento de pacientes.

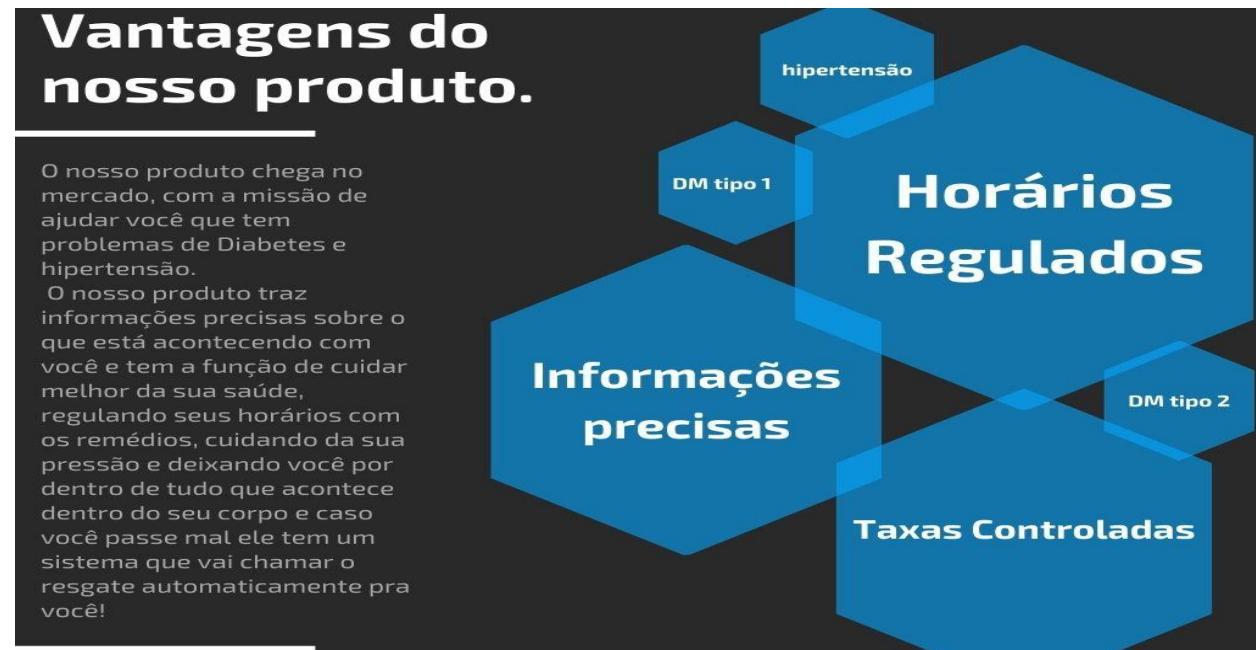
A produtividade dependerá de vários fatores, como a precisão das medições, a confiabilidade do hardware e software, a facilidade de uso e integração com outros sistemas hospitalares, entre outros.

Assumindo que, seja confiável e preciso, pode-se esperar que ele aumente significativamente a eficiência dos processos de monitorização de pacientes. Por exemplo, em vez de um enfermeiro ter que medir a pressão arterial de um paciente várias vezes ao dia, o robô pode realizar as medições automaticamente e enviar os dados para o prontuário eletrônico do paciente em tempo real. Isso reduziria a carga de trabalho dos enfermeiros e médicos, permitindo que eles se concentrem em outras tarefas importantes.

Além disso, o robô pode ajudar a identificar precocemente situações críticas, alertando o enfermeiro ou médico quando os níveis de pressão arterial ou açúcar no sangue estão acima do normal. Isso pode levar a intervenções mais rápidas e eficazes, melhorando a qualidade do cuidado do paciente e reduzindo o tempo de internação.

Em termos de números, fazer uma estimativa precisa de produtividade dependerá do número de pacientes atendidos, da frequência das medições, e outros fatores específicos do hospital.

Se o robô for implantado com sucesso em um ambiente hospitalar, pode-se esperar que ele contribua significativamente para a eficiência dos processos de monitorização de pacientes e melhore a qualidade do cuidado oferecido.



Supondo que o hospital tenha 50 pacientes, o robô pode ser programado para medir a pressão arterial e os níveis de açúcar no sangue desses pacientes a cada 2 ou 4 horas, dependendo das necessidades específicas

de cada paciente. Se assumirmos que cada medição leva cerca de 5 minutos para ser concluída, isso significa que o robô precisaria de cerca de 10 a 20 horas por dia para medir todos os pacientes com a frequência desejada.

No entanto, é importante lembrar que o robô pode realizar medições simultâneas em vários pacientes ao mesmo tempo, e pode fazê-las 24 horas por dia sem pausas para descanso, ao contrário de um ser humano. Além disso, o robô pode alertar os enfermeiros ou médicos imediatamente quando detectar uma leitura anormal, permitindo intervenções rápidas e eficazes.

Tudo isso significa que pode aumentar significativamente a eficiência dos processos de monitorização de pacientes em um ambiente hospitalar, permitindo que os enfermeiros e médicos se concentrem em outras tarefas importantes, enquanto o robô realiza as medições e envia alertas em tempo real. Isso pode levar a um atendimento mais eficiente e preciso, resultando em melhores resultados para os pacientes e redução dos custos hospitalares.



ORÇAMENTO DE PREÇOS E PRAZOS:

Geladeira Portátil: R\$626,00

Prazo de entrega: 9 dias úteis

Preço do frete: R\$12,90 para Cássia MG

(https://www.magazineluiza.com.br/mini-geladeira-multilaser-skin-care-espelhada-com-led-6-litros-12-110-220v-branca-tv017/p/ehh767fdd1/ed/mggp/?&seller_id=kingsomdistribuidora&utm_source=google&utm_medium=pla&utm_campaign=&partner_id=69993&gclid=CjwKCAjwov6hBhBsEiwAvrvN6A8C62lQ2sLqOJ5IT99WuRPNn_91be7FVtF9yayiu_m3WGmWQlPRKhoC2w0QAvD_BwE&gclsrc=aw.ds

Monitor lcd sensível ao toque de 7 polegadas

Prazo de entrega: 30 dias úteis.

Valor: 300,00

https://www.amazon.com.br/capacitiva-polegadas-compat%C3%ADvel-ferramenta-educacional/dp/B0BCFYWQHN?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&psc=1&smid=AFI5C2EF02M54

Bateria selada 12V 7AH

Prazo de entrega: 30 dias

Valor: 150,00

<https://a.co/d/2ncInH8>

Disco sólido interno western digital WD Green 240gb

Prazo de entrega: 7 Dias úteis

Valor: 160,00 (un)

<https://www.mercadolivre.com.br/disco-solido-interno-western-digital-wd-green-wds240g2g0b-240gb-verde/p/MLB8900210>

Adaptador Pci Express Tp-link Wn781nd 150mbps Rede Wi-fi

Prazo de entrega: 20 dias úteis

Valor: 115,00

https://produto.mercadolivre.com.br/MLB-3068746114-adaptador-pci-express-tp-link-wn-781nd-150mbps-rede-wi-fi-_JM

Oximeter Digital De Dedo Pulso Saturação De Oxigênio.

Prazo de entrega: 15 dias úteis

Valor: 52,00

https://produto.mercadolivre.com.br/MLB-3146703477-oximeter-digital-de-dedo-pulso-saturacao-de-oxignio-_JM?attributes=COLOR_SECONDARY_COLOR:QXp1bA

Sensor Infravermelho Reflexivo Industrial Ajustável Arduino

Prazo de entrega: 7 dias úteis

Valor: 53,00

https://produto.mercadolivre.com.br/MLB-2133165431-sensor-infravermelho-reflexivo-industrial-ajustavel-arduino-_JM

Sensor de proximidade

Prazo de entrega: 7 dias úteis

Valor: 35 reais (2un)

https://produto.mercadolivre.com.br/MLB-3143995436-2-sensor-de-obstaculo-ir-infravermelho-proximidade-arduino-_JM

Medidor de glicose

Prazo de entrega: 7 Dias úteis

Valor: 70,00

<https://a.co/d/b1oXoSb>

- Como funciona um medidor de pressão?

O aparelho de pressão funciona, basicamente, interrompendo o fluxo sanguíneo para que seja possível a leitura da pressão diastólica e sistólica do paciente. Na leitura analógica, feita com esfigmomanômetro e estetoscópio, a leitura é feita através do som

<https://www.mobiloc.com.br/blog/aparelho-de-pressao-digital/#:~:text=O%20aparelho%20de%20press%C3%A3o%20funciona,%C3%A9%20feita%20atrav%C3%A9s%20do%20som.>

- Como funciona um medidor de diabetes (glicosímetro)?

Os glicosímetros são compostos por uma fita reagente que entra em contato com um reflectômetro. Na maioria dos sistemas, a glicose do sangue capilar é oxidada para ácido glucônico e peróxido de hidrogênio após o contato do sangue nas fitas reagentes que contêm

glicose oxidase e peroxidase. Esta reação leva a uma alteração na cor da fita que pode ser interpretada pelo método fotométrico ou pelo método amperométrico.

https://www.saude.pr.gov.br/sites/default/arquivos_restritos/files/documento/2020-05/estudocaso_glicosimetro.pdf

FORMA DE ANÁLISE DOS RESULTADOS

Os resultados obtidos serão analisados e quantificados, para expor ao público, primeiramente local, a capacidade de um robô assistente na área. Os estudos serão publicados em revistas e apresentados em congressos da área de Tecnologia da Informação e saúde.

BANCO DE DADOS

Com o robô sendo autônomo é necessário um local que tenha armazenado sua programação de atividades, levando em consideração que serão feitas duas versões, doméstica e hospitalar, sabendo disso foi escolhido dois tipos de banco de dados: MongoDB para ambientes hospitalares e SQLite para ambientes domésticos.

MongoDB é um banco de dados NoSQL, termo utilizado para a ausência do SQL, que utiliza diversas linguagens de programação, JavaScript, Python, C, C++, PHP, sendo bastante útil quando se trata de grande quantidade de armazenamento de dados não estruturados. Utilizando como assistente em hospitais, além de auxiliar nos exames pré-definidos é possível armazenar dados, com rapidez e eficiência, dos pacientes como relatórios de exames, ficha médica e dentre outras informações pessoais.

Com o MongoDB os dados armazenados são localizados facilmente e são atualizados pelo próprio robô em tempo real.

SQLite é uma biblioteca em que se utiliza a linguagem C, implementada a uma base SQL. Sendo bastante flexível e fácil de armazenar informações, sendo leve e não levando muitos recursos de armazenamento sendo facilmente integrado a aplicações móveis. Além da programação pré-definida é possível armazenar uma grande quantidade de dados fornecidos e podendo atualizar em tempo real. Utilizando o sistema de transação ACID é altamente seguro e confiável, o que é crucial para aplicações médicas.

CÓDICO:

Código “main” do assistente virtual.

```
import threading
import time
import pygame
import speech_recognition as sr
import re

#toca arquivos de som
def toca_som(arquivo):
    pygame.mixer.init()
    pygame.mixer.music.load(arquivo)
    pygame.mixer.music.play()

def para_som():
    pygame.mixer.music.stop()

#reconhecimento de voz
def reconhecer_voz():
    rec = sr.Recognizer()

    # Loop para reconhecimento de voz contínuo
    while True:
        # Grava o áudio do microfone sem limite de tempo
        with sr.Microphone() as mic:
            rec.adjust_for_ambient_noise(mic)
            audio = rec.listen(mic, phrase_time_limit=5)

        try:
            # Usa o reconhecedor de voz para transcrever o áudio
            text = rec.recognize_google(audio, language='pt-BR')

            # Verifica se a transcrição não está vazia
            if text:
                text_corrigido = text.replace('depressão', 'de pressão').replace('expressão', 'pressão')
                # Exibe a transcrição e sai do loop
                print("O usuário disse: " + text_corrigido)
                return text_corrigido
        except sr.UnknownValueError:
            # Se não entender o que foi dito, continua o loop
```

```
        pass
    except sr.RequestError as e:
        print("Não foi possível conectar ao serviço de
reconhecimento de fala: {}".format(e))
        break

#medidor de pressão arterial
def medidor_teste_pressao():
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som,
args=('poderia_por_gentileza.mp3',))
    thread.start()
    time.sleep(1)
    print('\nPoderia, por gentileza, me informar acerca de sua pressão
arterial mais recente?')
    texto = reconhecer_voz()
    numeros = re.findall('\d+', texto)
    numeros_juntos = int(''.join(numeros))

    pressao_baixa = list(range(40, 106))

    pressao_boa_baixa = list(range(106, 113))

    pressao_otima = list(range(113, 129))

    pressao_normal = list(range(129, 138))

    prehiper = list(range(138, 141))

    hiper1 = list(range(141, 160))

    hiper2 = list(range(160, 180))

    hiper3 = list(range(180, 201))

    if numeros_juntos in pressao_baixa:
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('pressao_baixa.mp3',))
        thread.start()
```

```
time.sleep(1)
print('\nSua pressão arterial está baixa')
time.sleep(2)

elif numeros_juntos in pressao_boa_baixa:
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som,
args=('pressao_boa.mp3',))
    thread.start()
    time.sleep(1)
    print('\nSua pressão arterial está boa')
    time.sleep(2)

elif numeros_juntos in pressao_otima:
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som,
args=('pressao_otima.mp3',))
    thread.start()
    time.sleep(1)
    print('\nSua pressão arterial está ótima')
    time.sleep(2)

elif numeros_juntos in pressao_normal:
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som,
args=('pressao_normal.mp3',))
    thread.start()
    time.sleep(1)
    print('\nSua pressão arterial está normal')
    time.sleep(2)

elif numeros_juntos in prehiper:
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som,
args=('pre_hipertensao.mp3',))
    thread.start()
    time.sleep(1)
    print('\nSeus valores de pressão arterial são de
PRÉ-HIPERTENSÃO. É importante que você meça novamente e, se o valor
persistir, recomendamos que realize uma consulta médica para um
diagnóstico preciso.')
```

```

        time.sleep(10)
    elif numeros_juntos in hiper1:
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('hipertensao_estagio1.mp3',))
        thread.start()
        time.sleep(1)
        print('\nSeus valores de pressão arterial são de HIPERTENSÃO
ESTÁGIO 1. É importante que você meça novamente e, se o valor
persistir, recomendamos que realize uma consulta médica para um
diagnóstico preciso.')
        time.sleep(10)
    elif numeros_juntos in hiper2:
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('hipertensao_estagio2.mp3',))
        thread.start()
        time.sleep(1)
        print('\nSeus valores de pressão arterial são de HIPERTENSÃO
ESTÁGIO 2. É importante que você meça novamente e, se o valor
persistir, recomendamos que realize uma consulta médica para um
diagnóstico preciso.')
        time.sleep(10)
    elif numeros_juntos in hiper3:
        simres1 = ['sim', 'sim eu gostaria', 'sim sim', 'sim eu
preciso', 'preciso', 'sim eu gostaria', 'eu preciso de uma
ambulância', 'sim por favor']
        naores1 = ['não', 'não não', 'não preciso', 'não não preciso',
'não obrigado', 'não precisa']
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('hipertensao_estagio3.mp3',))
        thread.start()
        time.sleep(1)
        print('\nSeus valores de pressão arterial são de HIPERTENSÃO
ESTÁGIO 3. É importante que você meça novamente e, se o valor
persistir, recomendamos que realize uma consulta médica imediatamente
para um diagnóstico preciso. !Grave!')
        time.sleep(12)

```

```

def ambulancia_resposta():
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som,
args=('ambulancia.mp3',))
    thread.start()
    time.sleep(1)
    print('\nConsiderando os resultados obtidos de sua pressão
arterial, com sua permissão, podemos solicitar uma ambulância para o
caso de você estar impossibilitado de se dirigir ao hospital. Você
gostaria de uma ambulância?: ')
    textores1 = reconhecer_voz()
    textores1 = textores1.lower()
    if textores1 in simres1:
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('certo_ambulancia.mp3',))
        thread.start()
        time.sleep(1)
        print('\nCerto, espere no local. Uma ambulância está
se encaminhando para sua localização atual.')
        time.sleep(5)
    elif textores1 in naores1:
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('nao_ambulancia.mp3',))
        thread.start()
        time.sleep(1)
        print('\nOk, de certa forma, recomendamos que vá até
um hospital.')
        time.sleep(4)
    else:
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('desculpe_n_entendi.mp3',))
        thread.start()
        time.sleep(1)
        print('\nMe desculpe, não entendi')
        time.sleep(2)
        ambulancia_resposta()
        ambulancia_resposta()
else:

```

```

pygame.init()
para_som()
thread = threading.Thread(target=toca_som,
args=('desculpe_n_entendi.mp3',))
thread.start()
time.sleep(1)
print('\nMe desculpe, não entendi')
time.sleep(2)
medidor_teste_pressao()

#medidor de glicemia
def medidor_teste_glicemia():
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som,
args=('taxa_glicose.mp3',))
    thread.start()
    time.sleep(1)
    print('Taxa de glicose no sangue: ')
    glice = reconhecer_voz()
    numeros_glice = re.findall('\d+', glice)
    numeros_juntos_glice = int(''.join(numeros_glice))

    glicemia_normal = list(range(70, 100))
    prediabetes = list(range(100, 125))
    diabetes = list(range(125, 220))

    if numeros_juntos_glice in glicemia_normal:
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('glicose_normal.mp3',))
        thread.start()
        time.sleep(1)
        print('\nA taxa de glicose em seu sangue apresenta valores
normais.')
        time.sleep(3)
    elif numeros_juntos_glice in prediabetes:
        time.sleep(1)
        pygame.init()

```

```

        para_som()
        thread = threading.Thread(target=toca_som,
args=('pre_diabetes.mp3',))
        thread.start()
        time.sleep(1)
        print('\nA taxa de glicose em seu sangue apresenta valores de
PRÉ-DIABETES. É de suma importância assegurar que os procedimentos
para a realização do teste de glicose tenham sido seguidos
rigorosamente. O paciente deve realizar o teste em jejum por um
período mínimo de 8 horas. Ademais, recomendamos que busque uma
consulta médica a fim de obter um diagnóstico preciso.')
        time.sleep(18)
    elif numeros_juntos_glice in diabetes:
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('diabetes.mp3',))
        thread.start()
        time.sleep(1)
        print('\nA taxa de glicose em seu sangue apresenta valores de
DIABETES. É de suma importância assegurar que os procedimentos para a
realização do teste de glicose tenham sido seguidos rigorosamente. O
paciente deve realizar o teste em jejum por um período mínimo de 8
horas. Ademais, recomendamos que busque uma consulta médica a fim de
obter um diagnóstico preciso.')
        time.sleep(18)
    else:
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('glicose_invalida.mp3',))
        thread.start()
        time.sleep(1)
        print('\nNão foi possível identificar um valor de glicose
válido. Certifique-se de inserir apenas números.')
        time.sleep(4)
        medidor_teste_glicemia()

#Assistente virtual
def assistente_virtual():

```

```
time.sleep(1)
pygame.init()
para_som()
thread = threading.Thread(target=toca_som,
args=('assistente_virtual.mp3',))
thread.start()
time.sleep(1)
print('\nBoa tarde, sou o assistente virtual designado para auxiliá-lo durante todo o processo. Por gentileza, me informe se deseja realizar um teste específico de pressão sanguínea ou de glicemia, ou se prefere realizar ambos os testes.')
teste_especifico = ['teste específico', 'eu quero um teste específico', 'eu gostaria um teste específico', 'gostaria um teste específico', 'específico', 'um específico', 'quero um teste específico', 'preciso de um teste específico', 'gostaria de um teste específico', 'um teste específico por favor.', 'eu quero realizar um teste específico', 'quero realizar um teste específico']
teste_pressao = ['quero um teste de pressão', 'preciso medir minha pressão', 'gostaria de fazer um teste de pressão', 'teste de pressão', 'por favor verificar a minha pressão', 'teste de pressão', 'pressão', 'só o de pressão', 'teste de pressão arterial', 'pressão arterial', 'um teste de pressão arterial', 'um de pressão arterial', 'quero fazer um teste de pressão', 'fazer um teste de pressão', 'teste específico de pressão sanguínea', 'teste específico de pressão', 'quero fazer um teste específico de pressão sanguínea', 'de pressão', 'pressão sanguínea', 'teste de pressão sanguínea', 'o de pressão', 'o de pressão sanguínea', 'o teste de pressão', 'o teste de pressão sanguínea']
teste_glicemia = ['quero um teste de glicemia', 'preciso medir minha glicemia', 'gostaria de fazer um teste de glicemia', 'teste de glicemia', 'por favor verificar a minha glicemia', 'teste de glicemia', 'glicemia', 'só o de glicemia', 'teste de glicose', 'teste específico de glicemia', 'quero fazer um teste específico de glicemia',
```

```

        'de glicemia', 'glicose', 'o de glicemia', 'o de
glicose', 'o do teste de glicemia', 'o do teste de glicose']
    teste_ambos = ['para ambos', 'ambos', 'um teste para ambos', 'eu
quero um teste para os dois',
                    'eu quero um teste para ambos', 'eu quero um teste
dos dois', 'dos dois', 'ambos os testes',
                    'teste de ambos', 'o teste de ambos', 'o de ambos',
'o de ambos os testes', 'um teste de ambos', 'realizar ambos os
testes']

apresentacao_resposta1 = reconhecer_voz()
apresentacao_resposta1 = apresentacao_resposta1.lower()
if apresentacao_resposta1 in teste_especifico:
    def apresentacao_resposta2():
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som,
args=('especifico.mp3',))
        thread.start()
        time.sleep(1)
        print('\nPor gentileza, me informe qual teste gostaria de
realizar: o de pressão arterial ou o de glicemia?')
        apresentacao_resposta2 = reconhecer_voz()
        apresentacao_resposta2 = apresentacao_resposta2.lower()
        if apresentacao_resposta2 in teste_pressao:
            medidor_teste_pressao()
        elif apresentacao_resposta2 in teste_glicemia:
            medidor_teste_glicemia()
        else:
            time.sleep(1)
            pygame.init()
            para_som()
            thread = threading.Thread(target=toca_som,
args=('desculpe_n_entendi.mp3',))
            thread.start()
            time.sleep(1)
            print('\nNão entendi!')
            time.sleep(2)
            return apresentacao_resposta2()
        apresentacao_resposta2()
    elif apresentacao_resposta1 in teste_pressao:
        medidor_teste_pressao()
    elif apresentacao_resposta1 in teste_glicemia:
        medidor_teste_glicemia()

```

```

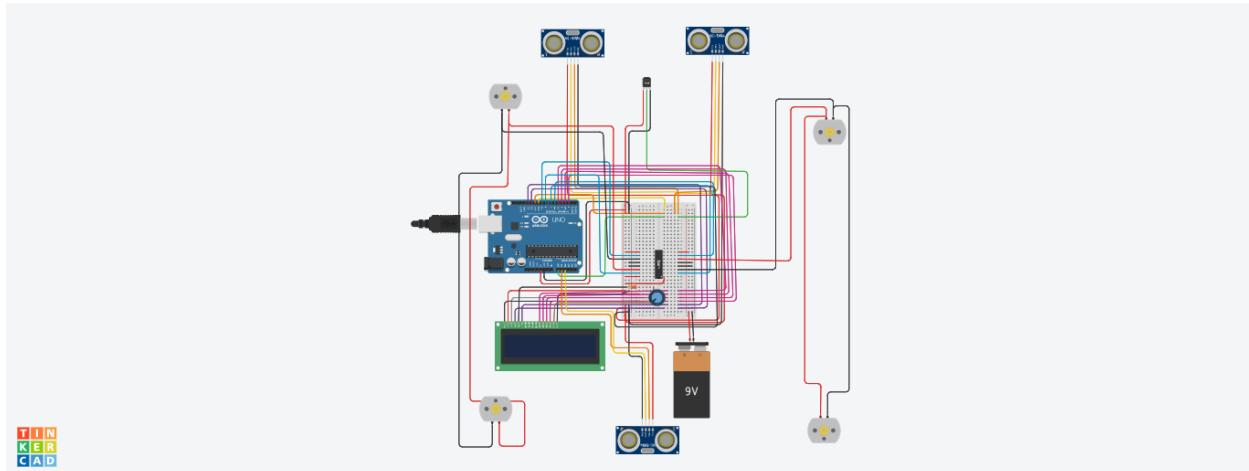
    elif apresentacao_resposta1 in teste_ambos:
        medidor_teste_pressao(), medidor_teste_glicemia()
    else:
        thread = threading.Thread(target=toca_som,
args=('desculpe_tente novamente.mp3',))
        thread.start()
        time.sleep(1)
        print('\nDesculpe, eu não comprehendi. Tente novamente.')
        time.sleep(2)
        return assistente_virtual()

assistente_virtual()

```

ROBÔ

Fiação base



Essa seria a fiação da movimentação do robô e como o programa usado não tem suporte para todos os processos necessários esse foi o máximo de comparações possíveis de se fazer

Código

```

// C++ code
//
#include <LiquidCrystal.h>

```

```
int segundos = 0;

int distance = 0;

int back = 0;

int temperatura = 0;

int precoao = 0;

int diabete = 0;

int base_de_dados_p = 0;

int base_de_dados_d = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
    pinMode(triggerPin, OUTPUT); // Clear the trigger
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    // Sets the trigger pin to HIGH state for 10 microseconds
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
    pinMode(echoPin, INPUT);
    // Reads the echo pin, and returns the sound wave travel time in microseconds
    return pulseIn(echoPin, HIGH);
}

LiquidCrystal lcd_1(12, 13, 5, 4, 3, 2);

int counter;

int counter2;

void setup()
{
    Serial.begin(9600);
    pinMode(7, OUTPUT);
    pinMode(8, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(9, OUTPUT);
    lcd_1.begin(16, 2); // Set up the number of columns and rows on the LCD.
```

```
}

void loop()
{
    distance = 0.01723 * readUltrasonicDistance(10, 11);
    if (distance > 25) {
        // frente
        Serial.println("forward ");
        digitalWrite(7, HIGH);
        digitalWrite(8, HIGH);
        digitalWrite(6, LOW);
        digitalWrite(9, LOW);
    } else {
        Serial.println("pare");
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        digitalWrite(6, LOW);
        digitalWrite(9, LOW);
        delay(2000); // Wait for 2000 millisecond(s)
        Serial.println("volta");
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        digitalWrite(6, HIGH);
        digitalWrite(9, HIGH);
        delay(2000); // Wait for 2000 millisecond(s)
        back = 0.01723 * readUltrasonicDistance(A1, A2);
        if (back > 25) {
            Serial.println("Right");
            digitalWrite(7, HIGH);
            digitalWrite(8, HIGH);
            digitalWrite(6, LOW);
            digitalWrite(9, LOW);
            delay(2000); // Wait for 2000 millisecond(s)
        } else {
            Serial.println("pare2");
            digitalWrite(7, LOW);
            digitalWrite(8, LOW);
            digitalWrite(6, LOW);
            digitalWrite(9, LOW);
            delay(2000); // Wait for 2000 millisecond(s)
            Serial.println("volta 2");
            digitalWrite(7, LOW);
            digitalWrite(8, LOW);
            digitalWrite(6, HIGH);
        }
    }
}
```

```
digitalWrite(9, HIGH);
delay(2000); // Wait for 2000 millisecond(s)
Serial.println("volta em u");
digitalWrite(7, HIGH);
digitalWrite(8, LOW);
digitalWrite(6, LOW);
digitalWrite(9, HIGH);
delay(3000); // Wait for 3000 millisecond(s)
}
}
pressao = 0;
lcd_1.print("iniciando ...");
for (counter = 0; counter < 8; ++counter) {
  lcd_1.setCursor(0, 1);
  lcd_1.print(segundos);
  delay(1000); // Wait for 1000 millisecond(s)
  segundos += 1;
}
lcd_1.clear();
lcd_1.print("carregando... ");
delay(2000); // Wait for 2000 millisecond(s)
lcd_1.clear();
base de dados p = 0;
// dependendo da base de dados o resultado pode
// mudar
lcd_1.print("resultado :");
lcd_1.clear();
diabete = 0;
lcd_1.print("iniciando...");
for (counter2 = 0; counter2 < 8; ++counter2) {
  lcd_1.setCursor(0, 1);
  lcd_1.print(segundos);
  delay(1000); // Wait for 1000 millisecond(s)
  segundos += 1;
}
lcd_1.clear();
lcd_1.print("carregando... ");
delay(2000); // Wait for 2000 millisecond(s)
lcd_1.clear();
base de dados d = 0;
lcd_1.print("resultado:");
lcd_1.clear();
}
```

ALGORITMO

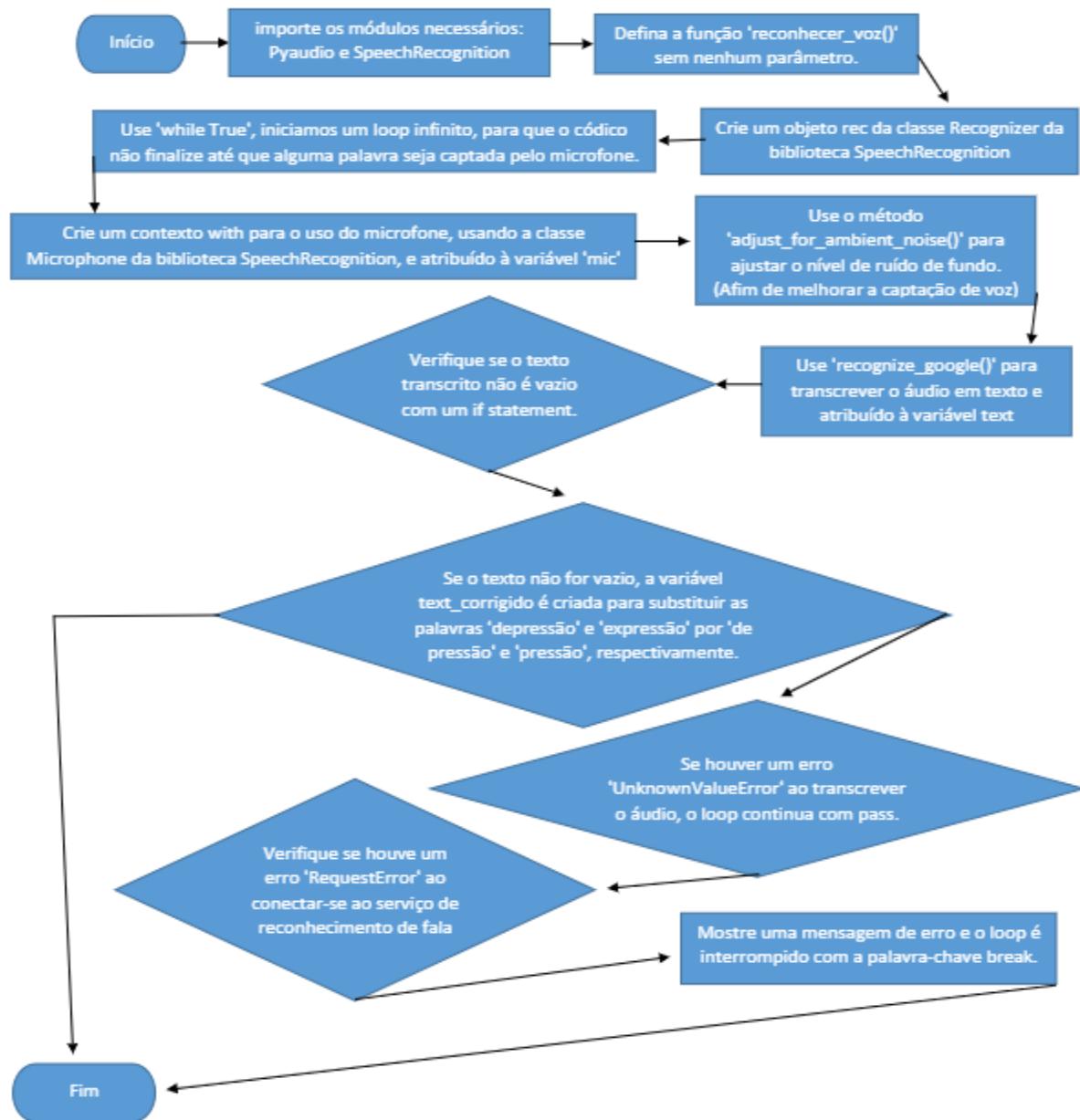
Obs: Os algoritmos de descrição narrativa e fluxograma foram feitos separados por causa das alterações que foram feitas no meio do desenvolvimento do código, então decidimos fazer cada etapa do código separadas e apenas o pseudocódigo feito com todos os programas juntos.

Descrição narrativa (RECONHECIMENTO_VOZ):

- 1: Primeiro, importamos os módulos necessários: Pyaudio e SpeechRecognition.
- 2: Definimos a função 'reconhecer_voz()' sem nenhum parâmetro.
- 3: Criamos um objeto rec da classe Recognizer da biblioteca SpeechRecognition.
- 4: Usando 'while True', iniciamos um loop infinito, para que o código não finalize até que alguma palavra seja captada pelo microfone.
- 5: É criado um contexto with para o uso do microfone, usando a classe Microphone da biblioteca SpeechRecognition, e atribuído à variável 'mic'.
- 6: Usamos o método 'adjust_for_ambient_noise()' para ajustar o nível de ruído de fundo. (Afim de melhorar a captação de voz)
- 7: É chamado o método 'listen()' para capturar o áudio do microfone por até 5 segundos e atribuído à variável 'audio'.
- 8: Usando 'recognize_google()' para transcrever o áudio em texto e atribuído à variável text, com a opção de linguagem 'pt-BR'.
- 9: Então, é verificado se o texto transcritó não é vazio com um if statement.
- 10: E, se o texto não for vazio, a variável text_corrigido é criada para substituir as palavras 'depressão' e 'expressão' por 'de pressão' e 'pressão', respectivamente. (por se tratar de um reconhecimento de voz que usa a internet, alguns fatores influenciam na precisão da captação de voz)
- 11: O texto transcritó corrigido é exibido na tela com a mensagem "O usuário disse: " concatenada com a string text_corrigido.
- 12: O texto transcritó corrigido é retornado pela função usando a declaração return.
- 13: E, se houver um erro 'UnknownValueError' ao transcrever o áudio, o loop continua com pass.

14: Por fim, se houver um erro 'RequestError' ao conectar-se ao serviço de reconhecimento de fala, uma mensagem de erro é exibida e o loop é interrompido com a palavra-chave break.

Fluxograma (RECONHECIMENTO_VOZ):



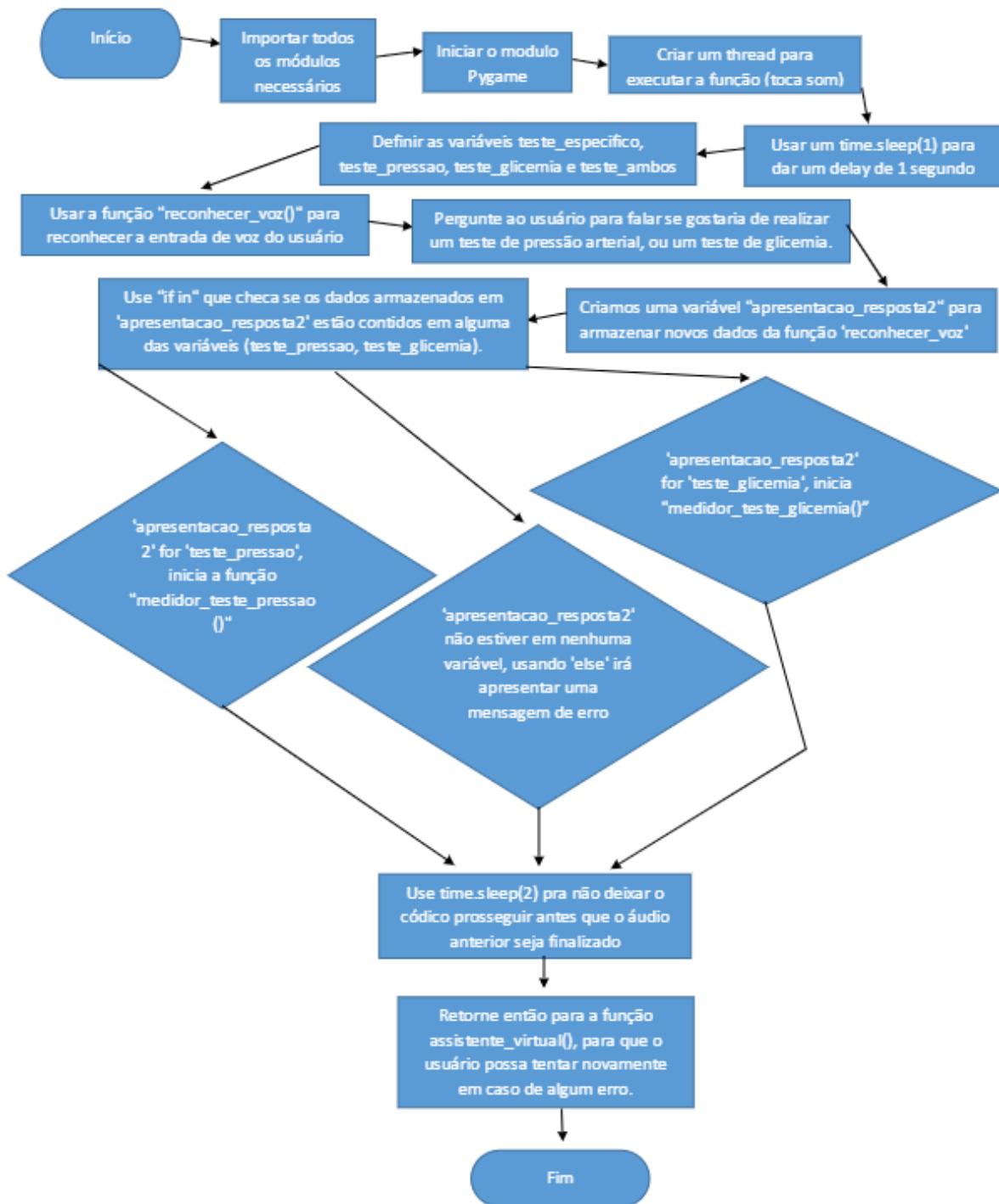
Descrição narrativa (Assistente_virtual):

Legenda: "*" é usado para códicos maiores que se repetem (no caso o sistema de parar a reprodução de um áudio, e para iniciar outro).

- 1: Primeiro, importamos todas os módulos necessários: threading, time, pygame, SpeechRecognition e re. (por se tratar do assistente virtual, que abrange todas as outras funções em seu funcionamento, ele precisa de todas as bibliotecas)
- 2: Definimos a função assistente_virtual()
- 3: Em seguida, iniciamos o módulo pygame.
- 4: Depois que o módulo pygame foi iniciado, chamamos a função 'para_som()' (para parar qualquer áudio que estiver sendo tocado).
- 5: Então, é criado uma thread para executar a função toca_som('assistente_virtual.mp3').
- 6: Inicia a thread criada. (a função thread em python, faz com que um código definido por você seja processado em segundo plano, permitindo uma continuidade no código, e não tendo que esperar o mesmo ser finalizado. Deixando o sistema mais flúido)
- 7: Usamos um time.sleep(1) para dar um delay de 1 segundo antes que o código continue, dando tempo suficiente para o arquivo de áudio da função 'thread' inicialize.
- 8: Em seguida, damos uma apresentação ao usuário usando 'print' (e o áudio que está sendo reproduzido em segundo plano), pedindo para que o usuário decida entre fazer um teste específico, ou ambos os testes.
- 9: Definimos as variáveis teste_especifico, teste_pressao, teste_glicemia e teste_ambos, que contêm as palavras-chave que o assistente virtual reconhece como comandos de teste específico de pressão, teste de pressão, teste de glicemia e ambos os testes, respectivamente.
- 10: Usamos a função "reconhecer_voz()" para reconhecer a entrada de voz do usuário e armazenamos o resultado na variável "apresentacao_resposta1".
- 11: Em seguida, convertemos a resposta em minúsculas usando 'lower()'. (isso faz com que toda palavra transcrita pelo reconhecimento de voz, seja escrita com letras minúsculas. Isso faz com que economizamos strings, deixando o código mais otimizado e rápido. Exemplo: 'sim' e 'Sim', representam a mesma coisa, mas sem 'lower()', teria que ter o dobro de variável).
- 12: Depois, verifica se a resposta do usuário está contida na lista 'teste_especifico'.
- 13: Se sim, define a função 'apresentacao_resposta22()'.
- 14: A função 'apresentacao_resposta22()' é criada então para usuários que responderam 'fazer um teste específico'.
- 15: Usamos um time.sleep(1) para dar um pequeno delay, afim de evitar erros, processando o código em ordem correta.
- 16: Então repetimos o código do início do 'assistente_virtual()'.
- 17*: Iniciamos o módulo pygame.
- 18*: Depois que o módulo pygame foi iniciado, chamamos a função 'para_som()' (para parar qualquer áudio que estiver sendo tocado).
- 19*: Então, é criado uma thread para executar a função toca_som('especifico.mp3').
- 20*: Iniciamos a thread criada.
- 21*: Usamos um time.sleep(1) para dar um delay de 1 segundo antes que o código continue, dando tempo suficiente para o arquivo de áudio da função 'thread' inicialize.
- 22: Em seguida, perguntamos ao usuário para falar se gostaria de realizar um teste de pressão arterial, ou um teste de glicemia.

- 23: Criamos uma variável "apresentacao_resposta2" para armazenar novos dados da função 'reconhecer_voz'
- 24: Usamos então, "if in" que checa se os dados armazenados em 'apresentacao_resposta2' estão contidos em alguma das variáveis (teste_pressao, teste_glicemias).
- 25: Se 'apresentacao_resposta2' estiver contido em 'teste_pressao', inicializa a função "medidor_teste_pressao()".
- 26: Se 'apresentacao_resposta2' estiver contido em 'teste_glicemias', inicializa a função "medidor_teste_glicemias()".
- 27: Se 'apresentacao_resposta2' não estiver contido em nenhuma variável, usando 'else' irá apresentar um mensagem de erro para o usuário e reproduzirá um áudio (USANDO NOVAMENTE OS MESMO CÓDICOS EM "*", MUDANDO APENAS O ARQUIVO DE SOM. NÃO IREI REPETIR O MESMO CÓDICO NOVAMENTE).
- 28: Retorna o usuário para a função "apresentacao_respostas22()" para que o mesmo possa tentar novamente.
- 29: Porém, se a resposta do usuário da primeira pergunta do assistente virtual, estiver contida em uma das variáveis (teste_pressao, teste_glicemias), a função medidor_teste_pressao(), e medidor_teste_glicemias(), são chamadas RESPECTIVAMENTE, usando 'elif'.
- 30: Caso a resposta do usuário não estiver contida em nenhuma de nossas variáveis, usando 'else', a função toca_som, reproduz um arquivo de som, juntamente com uma mensagem "Desculpe, eu não comprehendi. Tente novamente".
- 31: Um time.sleep(2) é usado por fim de não deixar o código prosseguir antes que o áudio anterior seja finalizado.
- 31: Retorna então para a função assistente_virtual(), para que o usuário possa tentar novamente em caso de algum erro.

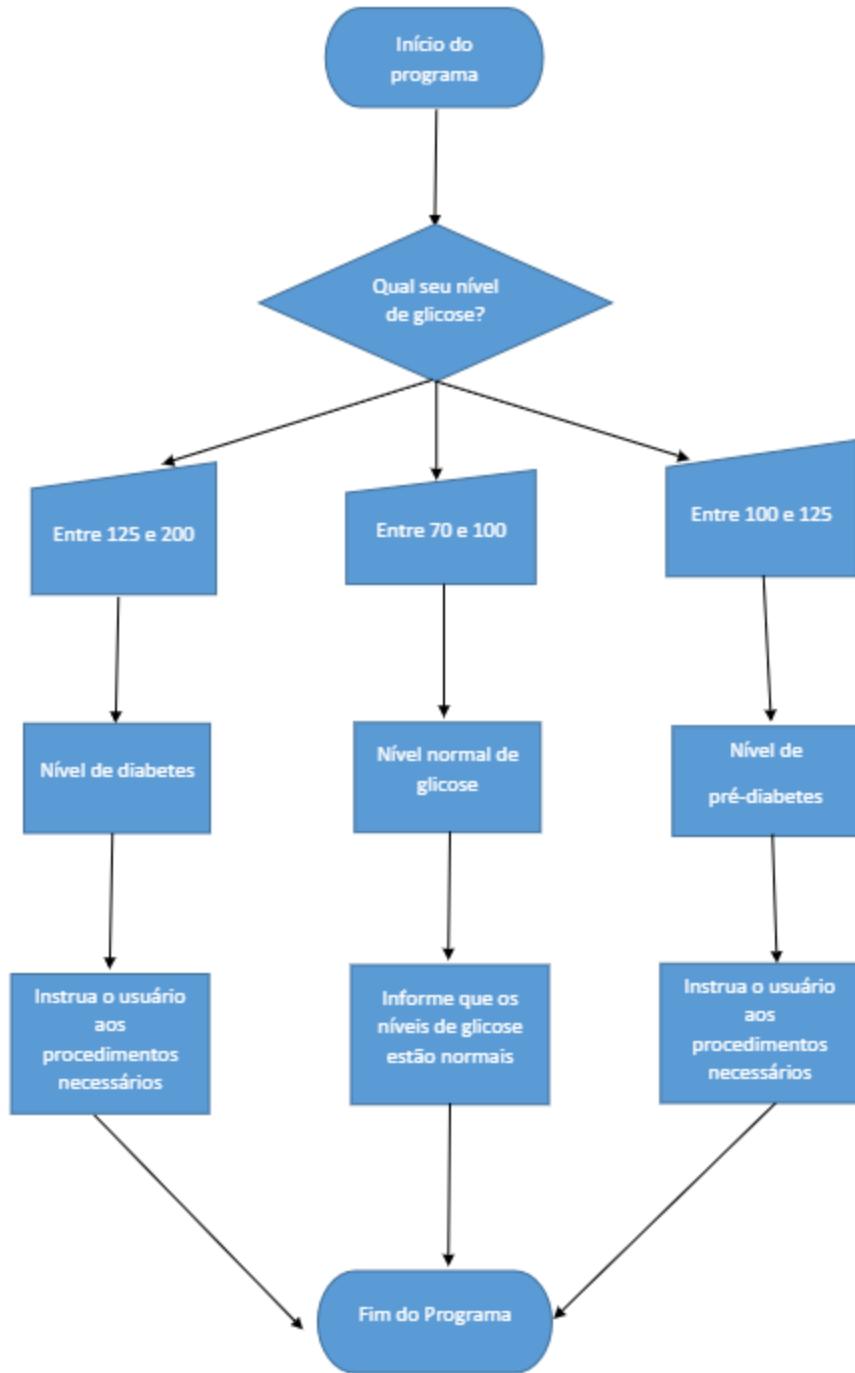
Fluxograma (Assistente_virtual):



Algoritmo descrição narrativa (TESTE_GLICEMIA):

- 1: Primeiro, importamos os módulos necessários: time, re, pygame e threading.
- 2: Definimos a função medidor_teste_glicemia()
- 3: Em seguida, chamamos a função "para_som()" para garantir que qualquer som anterior seja interrompido antes de começarmos a reproduzir o som atual.
- 4: Depois, iniciamos o módulo Pygame e criamos uma nova thread para tocar o arquivo de áudio "taxa_glicose.mp3", usando a função "toca_som()". (thread em python serve para rodar alguma função em segundo plano, normalmente algo mais pesado de ser processado)
- 5: Aguardamos um segundo para que o som comece a ser reproduzido. (usando o time.sleep(segundos))
- 6: Em seguida, pedimos ao usuário para inserir sua taxa de glicose no sangue.
- 7: Usamos a função "reconhecer_voz()" para reconhecer a entrada de voz do usuário e armazenamos o resultado na variável "glice".
- 8: Em seguida, usamos a expressão regular "re.findall()" para extrair apenas os números da string "glice" e armazenamos o resultado na lista "numeros_glice".
- 9: Convertamos a lista de números em uma string usando a função "join()" e a convertemos em um número inteiro usando a função "int()", armazenando o resultado na variável "numeros_juntos_glice".
- 10: Criamos três listas de números representando faixas normais de glicemia, pré-diabetes e diabetes, respectivamente.
- 11: Usando uma instrução condicional "if-elif-else", verificamos em qual faixa a taxa de glicose do usuário se encontra.
- 12: Se a taxa de glicose estiver na faixa normal, reproduzimos uma mensagem de áudio e imprimimos uma mensagem na tela informando que os valores de glicose estão normais.
- 13: Se a taxa de glicose estiver na faixa de pré-diabetes ou diabetes, reproduzimos uma mensagem de áudio e imprimimos uma mensagem na tela informando que os valores de glicose estão em níveis preocupantes, e recomendamos que o usuário consulte um médico.
- 14: Se a entrada do usuário não for uma taxa de glicose válida, reproduzimos uma mensagem de áudio e imprimimos uma mensagem na tela informando que a entrada não é válida e solicitando que o usuário insira apenas números.
- 15: Por fim, se houver um erro de entrada, chamamos a função "medidor_teste_glicemia()" novamente para reiniciar o processo de medição.

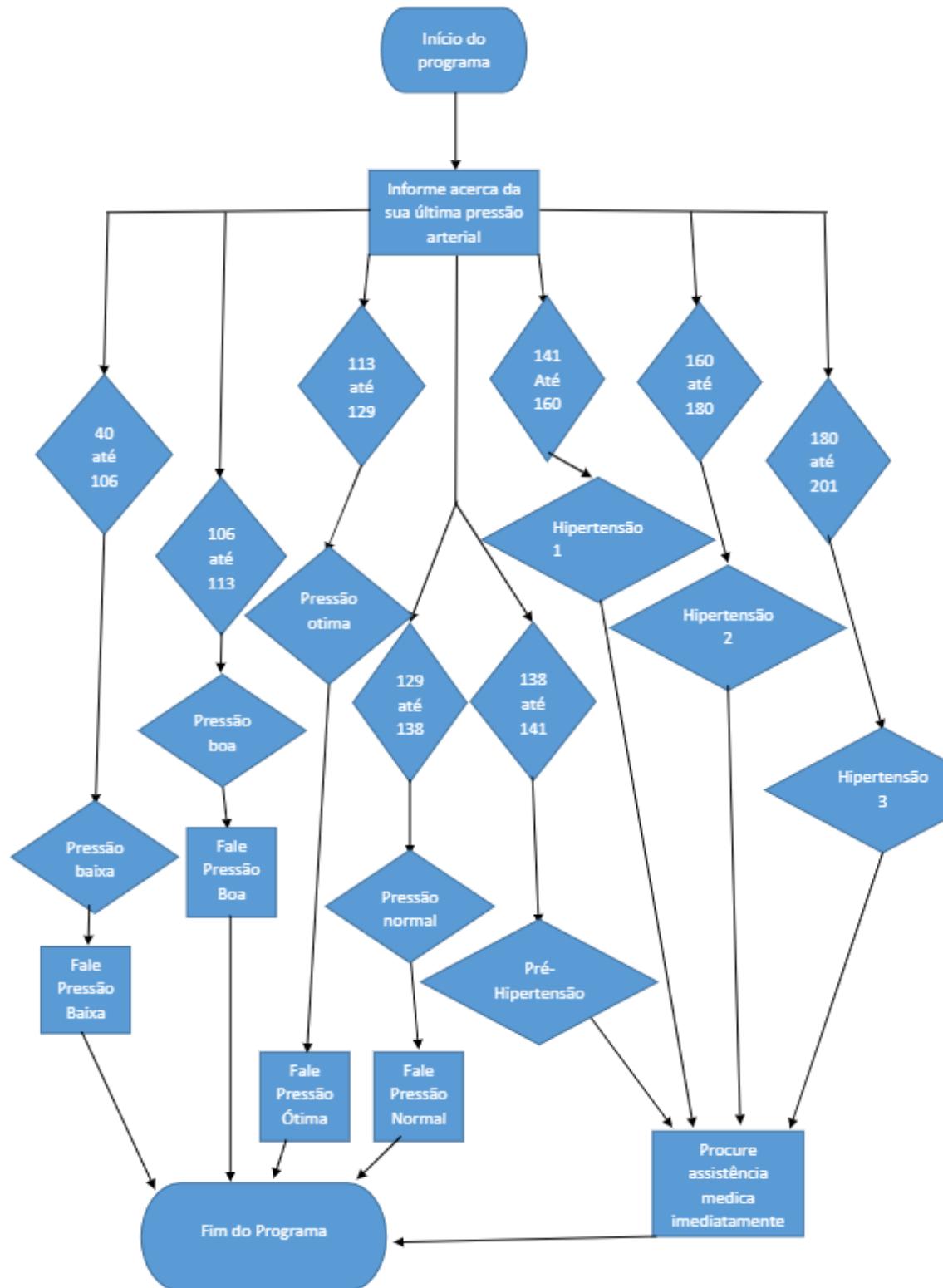
Fluxograma (TESTE_GLICEMIA):



Algoritmo descrição narrativa (Teste de pressão):

- 1: Primeiro, importamos os módulos necessários: time, re, pygame e threading.
- 2: Definimos a função medidor_teste_pressão()
- 3: Em seguida, chamamos a função "para_som()" para garantir que qualquer som anterior seja interrompido antes de começarmos a reproduzir o som atual.
- 4: Depois, iniciamos o módulo Pygame e criamos uma nova thread para tocar o arquivo de áudio "poderia_por_gentileza.mp3", usando a função "toca_som()".
- 5: Aguardamos um segundo para que o som comece a ser reproduzido.
- 6: Em seguida, pedimos ao usuário para inserir sua pressão arterial mais recente.
- 7: Usamos a função "reconhecer_voz()" para reconhecer a entrada de voz do usuário e armazenamos o resultado na variável numeros_juntos.
- 10: Criamos três listas de números representando pressão baixa, pressão boa, pressão otima, pressão normal, pré-hipertenso, hipertensão 1, hipertensão 2 e hipertensão 3
- 11: Crie as variáveis para cada uma informando as condições se a pressão está baixa, boa, otima, normal, pré hiper, hiper 1, hiper 2 e hiper 3.
- 12: Caso seja informado pré hiper, hiper 1, hiper 2 ou hiper 3, solicite que o usuário procure atendimento médico de imediato.
- 13: Por fim, se houver um erro de entrada, chamamos a função "medidor_teste_pressão()" novamente para reiniciar o processo de medição.

Fluxograma (Teste de pressão):



Algoritmo descrição narrativa (TOCA E PARA SOM):

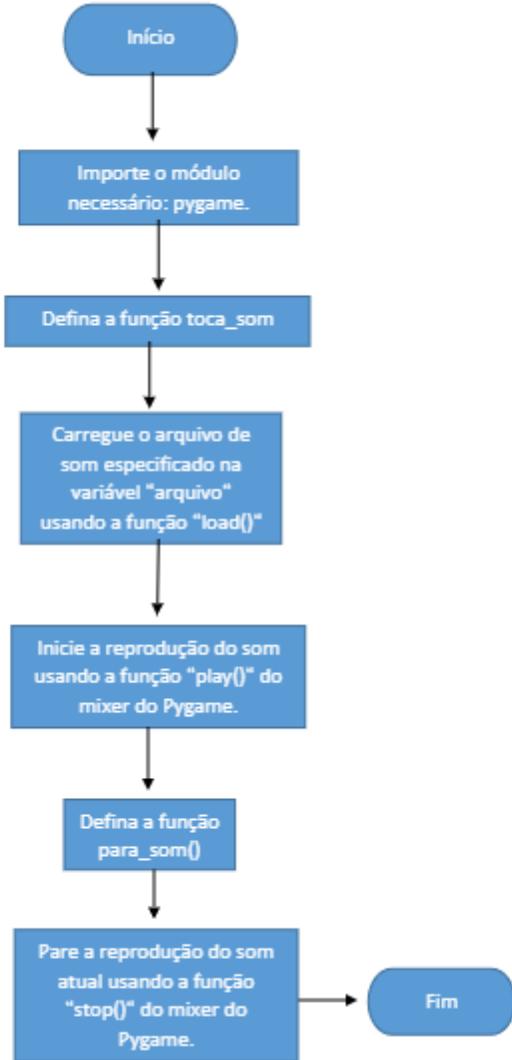
Função 'toca_som(arquivo)':

- 1: Primeiro, importamos o módulo necessário: pygame.
- 2: Definimos a função toca_som(arquivo)
- 3: Inicializamos o mixer do Pygame.
- 4: Carregamos o arquivo de som especificado na variável "arquivo" usando a função "load()" do mixer do Pygame.
- 5: Depois, iniciamos a reprodução do som usando a função "play()" do mixer do Pygame.

Função 'para_som()':

- 1: Definimos a função para_som()
- 2: Pare a reprodução do som atual usando a função "stop()" do mixer do Pygame.

Fluxograma (TOCA E PARA SOM):



Pseudocódigo:

```
//importe as bibliotecas
threading
time
pygame
speech_recognition as sr
re

//defina def toca_som(arquivo)
    pygame.mixer.init()
    pygame.mixer.music.load(arquivo)
    pygame.mixer.music.play()

//defina def para_som()
    pygame.mixer.music.stop()

//defina def reconhecer_voz()
    rec = sr.Recognizer()

//defina para loop
    while True
        with sr.Microphone() as mic
            rec.adjust_for_ambient_noise(mic)
            audio = rec.listen(mic, phrase_time_limit=5)

        try
            //defina para passar o que for falado para formato text
            text = rec.recognize_google(audio, language='pt-BR')

//defina para verificar se não há erro
    if text
        text_corrigido = text.replace('depressão', 'de pressão').replace('expressão', 'pressão')
        # Exibe a transcrição e sai do loop
        print(O usuário disse + text_corrigido)
        return text_corrigido
    except sr.UnknownValueError
//defina para continuar o loop do código
```

```

    pass
except sr.RequestError as e
    print(Não foi possível conectar ao serviço de reconhecimento de fala {0}.format(e))
    break

//defina para o medidor de pressão

medidor_teste_pressao()
time.sleep(1)
pygame.init()
para_som()
thread = threading.Thread(target=toca_som, args=('poderia_por_gentileza.mp3',))
thread.start()
time.sleep(1)
print('nPoderia, por gentileza, me informar acerca de sua pressão arterial mais recente')
texto = reconhecer_voz()
numeros = re.findall('d+', texto)
numeros_juntos = int("".join(numeros))

// Defina as variaveis

pressao_baixa = list(range(40, 106))

pressao_boa_baixa = list(range(106, 113))

pressao_otima = list(range(113, 129))

pressao_normal = list(range(129, 138))

prehiper = list(range(138, 141))

hiper1 = list(range(141, 160))

hiper2 = list(range(160, 180))

hiper3 = list(range(180, 201))

//defina as variáveis para cada situação

```

```
//Se numeros_juntos in pressao_baixa
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('pressao_baixa.mp3',))
    thread.start()
    time.sleep(1)
    print('nSua pressão arterial está baixa')
    time.sleep(2)
//Se não se numeros_juntos in pressao_boa_baixa
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('pressao_boa.mp3',))
    thread.start()
    time.sleep(1)
    print('nSua pressão arterial está boa')
    time.sleep(2)
//Se não se numeros_juntos in pressao_otima
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('pressao_otima.mp3',))
    thread.start()
    time.sleep(1)
    print('nSua pressão arterial está ótima')
    time.sleep(2)
//Se não se numeros_juntos in pressao_normal
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('pressao_normal.mp3',))
    thread.start()
    time.sleep(1)
    print('nSua pressão arterial está normal')
    time.sleep(2)
//Se não se numeros_juntos in prehiper
```

```

para_som()
thread = threading.Thread(target=toca_som, args=('pre_hipertensao.mp3',))
thread.start()
time.sleep(1)
print('nSeus valores de pressão arterial são de PRÉ-HIPERTENSÃO. É importante que você
meça novamente e, se o valor persistir, recomendamos que realize uma consulta médica para um
diagnóstico preciso.')
time.sleep(10)
//Se não se numeros_juntos in hiper1
time.sleep(1)
pygame.init()
para_som()
thread = threading.Thread(target=toca_som, args='hipertensao_estagio1.mp3')
thread.start()
time.sleep(1)
print('nSeus valores de pressão arterial são de HIPERTENSÃO ESTÁGIO 1. É importante
que você meça novamente e, se o valor persistir, recomendamos que realize uma consulta médica
para um diagnóstico preciso.')
time.sleep(10)
//Se não se numeros_juntos in hiper2
time.sleep(1)
pygame.init()
para_som()
thread = threading.Thread(target=toca_som, args='hipertensao_estagio2.mp3')
thread.start()
time.sleep(1)
print('nSeus valores de pressão arterial são de HIPERTENSÃO ESTÁGIO 2. É importante
que você meça novamente e, se o valor persistir, recomendamos que realize uma consulta médica
para um diagnóstico preciso.')
time.sleep(10)
//Se não se numeros_juntos in hiper3
simres1 = ['sim', 'sim eu gostaria', 'sim sim', 'sim eu preciso', 'preciso', 'sim eu gostaria', 'eu
preciso de uma ambulância', 'sim por favor']
naores1 = ['não', 'não não', 'não preciso', 'não não preciso', 'não obrigado', 'não precisa']
time.sleep(1)
pygame.init()
para_som()
thread = threading.Thread(target=toca_som, args='hipertensao_estagio3.mp3')
thread.start()
time.sleep(1)

```

```

print('nSeus valores de pressão arterial são de HIPERTENSÃO ESTÁGIO 3. É importante
que você meça novamente e, se o valor persistir, recomendamos que realize uma consulta médica
imediatamente para um diagnóstico preciso. !Grave!')
time.sleep(12)
def ambulancia_resposta()
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('ambulancia.mp3',))
    thread.start()
    time.sleep(1)
    print('nConsiderando os resultados obtidos de sua pressão arterial, com sua permissão,
podemos solicitar uma ambulância para o caso de você estar impossibilitado de se dirigir ao
hospital. Você gostaria de uma ambulância ')
    textores1 = reconhecer_voz()
    textores1 = textores1.lower()
    //Se textores1 in simres1
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som, args=('certo_ambulancia.mp3',))
        thread.start()
        time.sleep(1)
        print('nCerto, espere no local. Uma ambulância está se encaminhando para sua
localização atual.')
        time.sleep(5)
    //Se não se textores1 in naores1
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som, args=('nao_ambulancia.mp3',))
        thread.start()
        time.sleep(1)
        print('nOk, de certa forma, recomendamos que vá até um hospital.')
        time.sleep(4)
    //Se Não
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som, args=('desculpe_n_entendi.mp3',))
        thread.start()
        time.sleep(1)
        print('nMe desculpe, não entendi')
        time.sleep(2)

```

```

ambulancia_resposta()
ambulancia_resposta()
//Se não
pygame.init()
para_som()
thread = threading.Thread(target=toca_som, args=('desculpe_n_entendi.mp3'))
thread.start()
time.sleep(1)
print('nMe desculpe, não entendi')
time.sleep(2)
medidor_teste_pressao()

//Defina para iniciar o medidor de pressão
def medidor_teste_glicemia()
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('taxa_glicose.mp3'))
    thread.start()
    time.sleep(1)
    print('Taxa de glicose no sangue ')
    glice = reconhecer_voz()
    numeros_glice = re.findall('d+', glice)
    numeros_juntos_glice = int("".join(numeros_glice))

    glicemia_normal = list(range(70, 100))
    prediabetes = list(range(100, 125))
    diabetes = list(range(125, 220))

//Se
    numeros_juntos_glice in glicemia_normal
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args='glicose_normal.mp3')
    thread.start()
    time.sleep(1)
    print('A taxa de glicose em seu sangue apresenta valores normais.')

```

```
time.sleep(3)
//Se não se
    numeros_juntos_glice in prediabetes
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('pre_diabetes.mp3',))
    thread.start()
    time.sleep(1)

    print('nA taxa de glicose em seu sangue apresenta valores de PRÉ-DIABETES. É de suma
importância assegurar que os procedimentos para a realização do teste de glicose tenham sido
seguidos rigorosamente. O paciente deve realizar o teste em jejum por um período mínimo de 8
horas. Ademais, recomendamos que busque uma consulta médica a fim de obter um diagnóstico
preciso.')
    time.sleep(18)
//Se não se
    numeros_juntos_glice in diabetes
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('diabetes.mp3',))
    thread.start()
    time.sleep(1)

    print('nA taxa de glicose em seu sangue apresenta valores de DIABETES. É de suma
importância assegurar que os procedimentos para a realização do teste de glicose tenham sido
seguidos rigorosamente. O paciente deve realizar o teste em jejum por um período mínimo de 8
horas. Ademais, recomendamos que busque uma consulta médica a fim de obter um diagnóstico
preciso.')
    time.sleep(18)
//Se não
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('glicose_invalida.mp3',))
    thread.start()
    time.sleep(1)

    print('nNão foi possível identificar um valor de glicose válido. Certifique-se de inserir
apenas números.')
    time.sleep(4)
    medidor_teste_glicemia()
```

```
// Defina para iniciar a assistente virtual
def assistente_virtual()
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('assistente_virtual.mp3',))
    thread.start()
    time.sleep(1)
```

print('nBoa tarde, sou o assistente virtual designado para auxiliá-lo durante todo o processo.
Por gentileza, me informe se deseja realizar um teste específico de pressão sanguínea ou de glicemia, ou se prefere realizar ambos os testes.')

//Defina as variaveis

```
teste_especifico = ['teste específico', 'eu quero um teste específico', 'eu gostaria um teste específico',
                     'gostaria um teste específico', 'específico', 'um específico', 'quero um teste específico',
                     'preciso de um teste específico', 'gostaria de um teste específico',
                     'um teste específico por favor.', 'eu quero realizar um teste específico',
                     'quero realizar um teste específico']

teste_pressao = ['quero um teste de pressão', 'preciso medir minha pressão',
                  'gostaria de fazer um teste de pressão', 'teste de pressão', 'por favor verificar a minha pressão',
                  'teste de pressão', 'pressão', 'só o de pressão', 'teste de pressão arterial', 'pressão arterial',
                  'um teste de pressão arterial', 'um de pressão arterial', 'quero fazer um teste de pressão',
                  'fazer um teste de pressão', 'teste específico de pressão sanguínea',
                  'teste específico de pressão', 'quero fazer um teste específico de pressão',
                  'quero fazer um teste específico de pressão sanguínea', 'de pressão', 'pressão sanguínea',
                  'teste de pressão sanguínea', 'o de pressão', 'o de pressão sanguínea', 'o teste de pressão',
                  'o teste de pressão sanguínea']

teste_glicemia = ['quero um teste de glicemia', 'preciso medir minha glicemia',
                  'gostaria de fazer um teste de glicemia', 'teste de glicemia',
```

```

'por favor verificar a minha glicemia', 'teste de glicemia', 'glicemia', 'só o de
glicemia',
    'teste de glicose', 'teste específico de glicemia', 'quero fazer um teste específico de
glicemia',
        'de glicemia', 'glicose', 'o de glicemia', 'o de glicose', 'o do teste de glicemia', 'o do
teste de glicose']
teste_ambos = ['para ambos', 'ambos', 'um teste para ambos', 'eu quero um teste para os dois',
    'eu quero um teste para ambos', 'eu quero um teste dos dois', 'dos dois', 'ambos os
testes',
        'teste de ambos', 'o teste de ambos', 'o de ambos', 'o de ambos os testes', 'um teste de
ambos', 'realizar ambos os testes']
//Defina para o reconhecimento de voz
apresentacao_resposta1 = reconhecer_voz()
apresentacao_resposta1 = apresentacao_resposta1.lower()
//Se apresentacao_resposta1 in teste_especifico
def apresentacao_resposta22():
    time.sleep(1)
    pygame.init()
    para_som()
    thread = threading.Thread(target=toca_som, args=('especifico.mp3',))
    thread.start()
    time.sleep(1)
    print('Por gentileza, me informe qual teste gostaria de realizar o de pressão arterial ou o
de glicemia')
    apresentacao_resposta2 = reconhecer_voz()
    apresentacao_resposta2 = apresentacao_resposta2.lower()
    //Se apresentacao_resposta2 in teste_pressao
        medidor_teste_pressao()
    //Se não se apresentacao_resposta2 in teste_glicemia
        medidor_teste_glicemia()
    //Se não
        time.sleep(1)
        pygame.init()
        para_som()
        thread = threading.Thread(target=toca_som, args=('desculpe_n_entendi.mp3',))
        thread.start()
        time.sleep(1)
        print('Não entendi!')
        time.sleep(2)
    return apresentacao_resposta22()

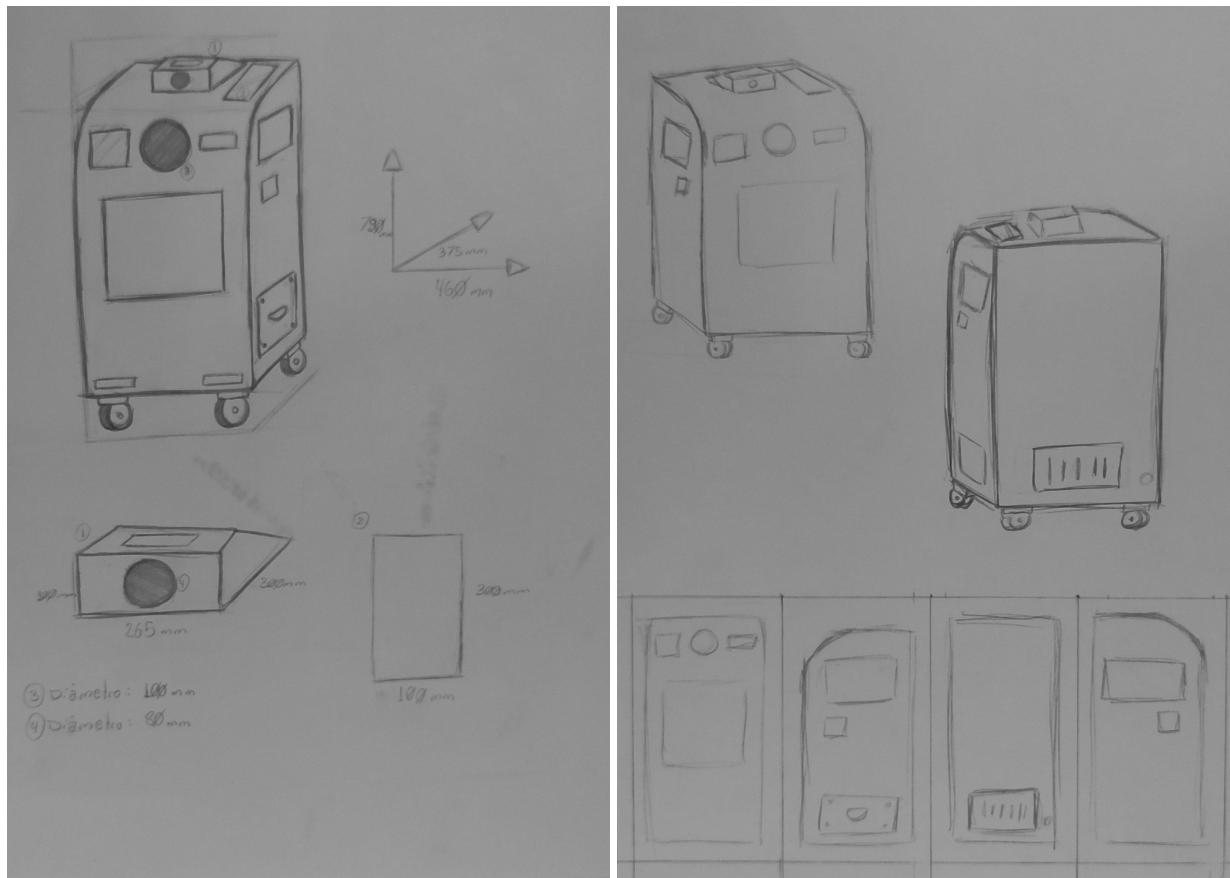
```

```
apresentacao_resposta22()
//Se não se apresentacao_resposta1 in teste_pressao
    medidor_teste_pressao()
//Se não se apresentacao_resposta1 in teste_glicemia
    medidor_teste_glicemia()
//Se não se apresentacao_resposta1 in teste_ambos
    medidor_teste_pressao(), medidor_teste_glicemia()
//Se não
    thread = threading.Thread(target=toca_som, args=('desculpe_tente novamente.mp3',))
    thread.start()
    time.sleep(1)
    print('nDesculpe, eu não comprehendi. Tente novamente.')
    time.sleep(2)
    return assistente_virtual()

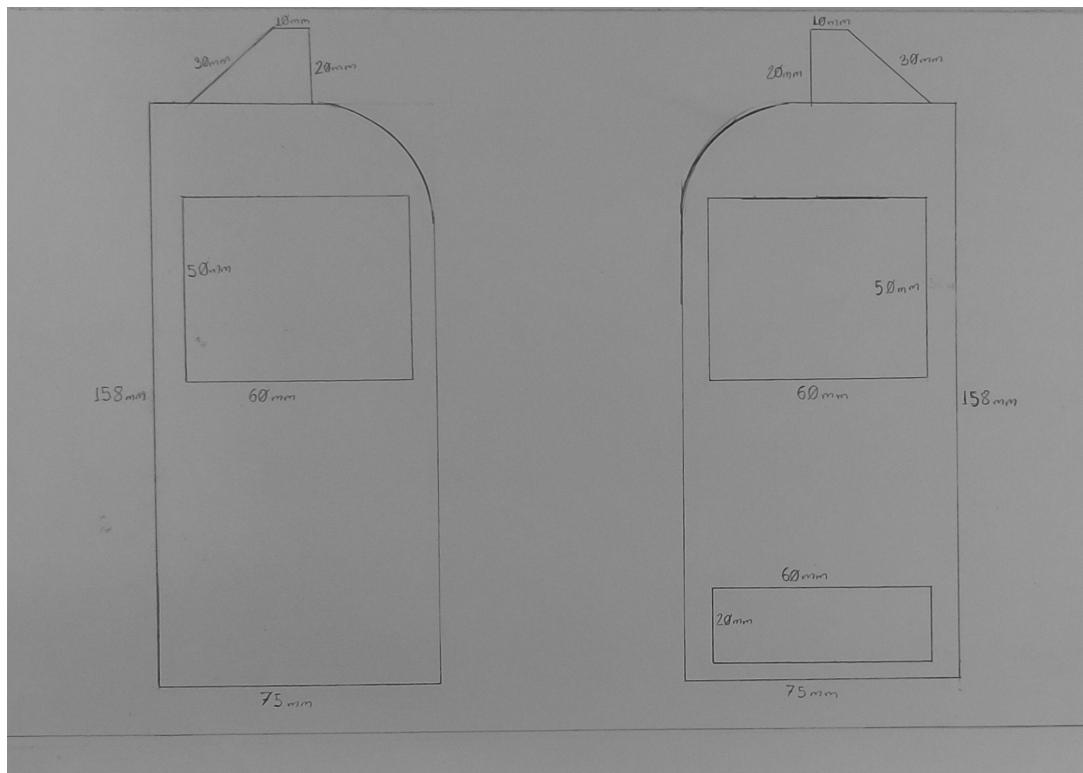
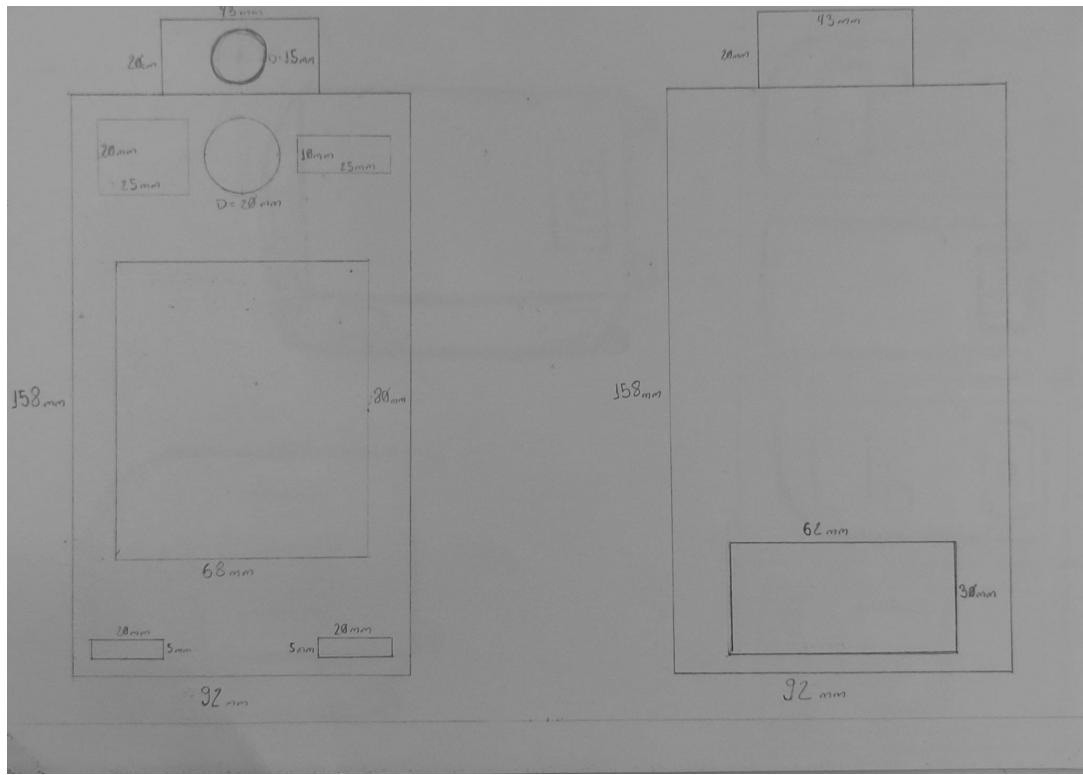
assistente_virtual()
```

DESIGN

Rascunho:



Desenho Técnico:



REFERÊNCIAS

Gross JL, Silveira SP, Camargo JL, Reichelt AJ, Azevedo MJ. **Diabetes Melito: Diagnóstico, Classificação e Avaliação do Controle Glicêmico.** Arq Bras Endocrinol Metab. 2002; 46(1). Acesso em: 28 de abril de 2023.

WESCHENFELDER MAGRINI, D.; GUE MARTINI, J. **Hipertensão arterial: principais fatores de risco modificáveis na estratégia de saúde da família.** Enfermería Global. v.11,n.26,p.344-353,abril de 2012. Disponível em:<http://scielo.isciii.es/pdf/eg/v11n26/pt_revision5.pdf>. Acesso em:. 28 de abril de 2023.

Referências usadas na criação do código

Teste de pressão arterial: de autoria minha (Mateus Lima Cançado)

Teste de glicemia: de autoria minha (Mateus Lima Cançado)

Assistente virtual: de autoria minha (Mateus Lima Cançado)

Reconhecimento de voz: Usei um vídeo para aprender (<https://youtu.be/fMyzSrDoT-E>)

E aperfeiçoei o código ao decorrer das semanas. Usei Chat GPT para aprender a fazer um comando de loop (while true) presente no código do reconhecimento de voz.

Funções para reproduzir áudios, parar de reproduzir: Chat GPT (eu estava usando a biblioteca playsound, fazendo do jeito simples, usando meu conhecimento. Porém para aperfeiçoar, usei o Chat GPT para aprender a fazer um sistema de som aprimorado)

Thread: Descobri essa função muito útil. Serve para executar algum comando em segundo plano, dando continuidade no restante do código.

(https://www.datacamp.com/tutorial/threading-in-python?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720824&utm_adgroupid=143216588537&utm_device=m&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=655068781134&utm_targetid=dsa-1947282172981&utm_loc_interest_ms=&utm_loc_physical_ms=9100596&utm_content=dsa~page~community-tuto&utm_campaign=230119_1-sea~dsa~tutorials_2-b2c_3-row-p2_4-prc_5-na_6-na_7-le_8-pdsh-g_o_9-na_10-na_11-na-aprfs23)

Para Fazer o algoritmo usei os conhecimentos básicos do início do curso do Projeto Fundação Bradesco:

Fundamentos de Lógica de Programação

(<https://www.ev.org.br/cursos/fundamentos-de-logica-de-programacao>)

Referências usadas na criação do código do robô

Livros:

Arduino Projects with Tinkercad

Arduino Projects with Tinkercad | Part 2: Design & program advanced Arduino-based electronics projects with Tinkercad

